Dr. Dietrich Boles, Matthias Mertens

# Hamster-Simulator-Plugin für das Eclipse SDK Version 1.0

## Benutzerhandbuch

**01.08.2006**

# Inhaltsverzeichnis


**1 Einleitung** **1**
- 1.1 Das Hamster Modell . . . 1
- 1.2 Der Hamster Simulator . . . 2
- 1.3 Änderungen gegenüber des Hamster-Simulators . . . 2

**2 Installation und Starten des Hamster-Simulator-Plugins** **6**
- 2.1 Vorraussetzungen . . . 6
- 2.2 Installation des Hamster-Simluator-Plugins . . . 6
- 2.3 Starten des Hamster-Simluator-Plugins . . . 7

**3 Ihr erstes Hamster-Programm** **9**
- 3.1 Erstellen eines Hamster-Projekts . . . 9
- 3.2 Gestalten eines Hamster-Territoriums . . . 10
- 3.3 Eingeben eines Hamster-Programms . . . 12
- 3.4 Ausführen eines Hamster-Programms . . . 14
- 3.5 Debuggen eines Hamster-Programms . . . 15
- 3.6 Zusammenfassung . . . 17

**4 Bedienung des Hamster-Simulator-Plugins** **18**
- 4.1 Die Bestandteile des Hamster-Simulator-Plugins . . . 18
  - 4.1.1 Globale Toolbar . . . 18
  - 4.1.2 Die Hamster-Wizards . . . 19
  - 4.1.3 Der Navigator View . . . 19
  - 4.1.4 Der Territorium-Editor . . . 20
  - 4.1.5 Der Hamster-Editor . . . 22
  - 4.1.6 Der Outline View . . . 23
  - 4.1.7 Der Problems View . . . 24
  - 4.1.8 Debug View . . . 25
  - 4.1.9 Variables View . . . 26
- 4.2 Verwalten von Hamster-Projekten . . . 26
  - 4.2.1 Erzeugen von Hamster-Projekten . . . 27
  - 4.2.2 Schließen und Öffnen von Hamster-Projekten . . . 28
  - 4.2.3 Unterordner in Hamster-Projekten . . . 28
  - 4.2.4 Probleme mit Hamster-Projekten . . . 29
- 4.3 Verwalten und Editieren von Hamster-Dateien . . . 30
  - 4.3.1 Schreiben eines neuen Hamster-Programms . . . 30
  - 4.3.2 Ändern eines existierenden Hamster-Programms . . . 31
  - 4.3.3 Speichern eines Hamster-Programms . . . 32
  - 4.3.4 Löschen eines existierenden Hamster-Programms . . . 33

4.3.5 Umbenennen eines existierenden Hamster-Programms . . . 33
4.3.6 Verschieben eines existierenden Hamster-Programms . . . 33
4.3.7 Kopieren eines existierenden Hamster-Programms . . . 34
4.3.8 Drucken eines Hamster-Programms . . . 34
4.3.9 Schließen eines geöffneten Hamster-Programms . . . 34
4.3.10 Editor-Funktionen . . . 35
4.4 Verwalten und Editieren von Hamster-Territorien . . . 40
4.4.1 Verändern der Größe des Hamster-Territoriums . . . 40
4.4.2 Platzieren des Standard-Hamsters im Hamster-Territorium . . . 41
4.4.3 Setzen der Blickrichtung des Standard-Hamsters . . . 41
4.4.4 Abfragen und Festlegen der Körneranzahl im Maul des Standard-Hamsters 41
4.4.5 Platzieren von Körnern auf Kacheln des Hamster-Territorium . . . 42
4.4.6 Platzieren von Mauern auf Kacheln des Hamster-Territorium . . . 42
4.4.7 Löschen von Kacheln des Hamster-Territorium . . . 43
4.4.8 Abspeichern eines Hamster-Territoriums . . . 43
4.4.9 Wiederherstellen eines abgespeicherten Hamster-Territoriums . . . 44
4.4.10 Umbenennen eines abgespeicherten Hamster-Territoriums . . . 44
4.4.11 Löschen und Verschieben und Kopieren von Hamster-Territorien . . . 45
4.4.12 Verändern der Größendarstellung des Hamster-Territoriums . . . 45
4.5 Kompilieren von Hamster-Programmen . . . 45
4.6 Ausführen von Hamster-Programmen . . . 45
4.7 Debuggen von Hamster-Programmen . . . 49
4.8 sonstige Funktionen des Hamster-Simulator-Pugins . . . 52
4.8.1 Umwandeln von Hamsterdateien ins „*.ham“-Format . . . 52
4.8.2 Umwandeln von „*.ham“-Dateien in Hamsterdateien . . . 52
4.8.3 Umwandeln von „*.cham“-Dateien in „*.oham“-Dateien . . . 53
4.8.4 Explizites Kompilieren von Hamsterdateien . . . 53
4.9 Sonstige Probleme . . . 53
4.9.1 Out of sync Fehler . . . 53

**5 Noch Fragen? 54**

**A Anhang 55**

**Abbildungsverzeichnis 58**

# 1 Einleitung

Programmieranfänger haben häufig Schwierigkeiten damit, dass sie beim Programmieren ihre normale Gedankenwelt verlassen und in eher technisch-orientierten Kategorien denken müssen, die ihnen von den Programmiersprachen vorgegeben werden. Gerade am Anfang strömen oft so viele inhaltliche und methodische Neuigkeiten auf sie ein, dass sie das Wesentliche der Programmierung, nämlich das Lösen von Problemen, aus den Augen verlieren. [1]

## 1.1 Das Hamster Modell

Das Hamster-Modell ist mit dem Ziel entwickelt worden, dieses Problem zu lösen. Mit dem Hamster-Modell wird Programmieranfängern ein einfaches, aber mächtiges Modell zur Verfügung gestellt, mit dessen Hilfe Grundkonzepte der imperativen und objektorientierten Programmierung auf spielerische Art und Weise erlernt werden können. Programmierer entwickeln so genannte Hamster-Programme, mit denen sie virtuelle Hamster durch eine virtuelle Landschaft steuern und bestimmte Aufgaben lösen lassen. Die Anzahl der gleichzeitig zu berücksichtigenden Konzepte wird im Hamster-Modell stark eingeschränkt und nach und nach erweitert. Prinzipiell ist das Hamster-Modell programmiersprachenunabhängig. Zum praktischen Umgang mit dem Modell wurde jedoch bewusst die Programmiersprache Java als Grundlage gewählt. Java - auch als „Sprache des Internet“ bezeichnet - ist eine moderne Programmiersprache, die sich in den letzten Jahren sowohl im Ausbildungsbereich als auch im industriellen Umfeld durchgesetzt hat. Zum Hamster-Modell existieren zwei Bücher. In dem ersten Buch „Programmieren spielend gelernt mit dem Java-Hamster-Modell“ werden allgemeine Grundlagen der Programmierung erläutert sowie Konzepte der imperativen Programmierung (Anweisungen, Schleifen, Prozeduren, Typen, Variablen, Parameter, Rekursion, ...) eingeführt. Darauf aufbauend behandelt das zweite Buch „Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell“ alle wichtigen Konzepte der objektorientierten Programmierung (Objekte, Klassen, Vererbung, Polymorphie, Interfaces, Exceptions, Zugriffsrechte, Pakete, ...). Beide Bücher sind insbesondere für Schüler und Studierende ohne Programmiererfahrung empfehlenswert. Für Programmierer mit Grundkenntnissen der imperativen Programmierung werden im zweiten Buch die Inhalte des ersten Buches kurz zusammengefasst. Die Bücher sind als Grundlage für Programmierkurse sowie zum Selbststudium geeignet. Dazu enthalten sie viele Beispielprogramme und Übungsaufgaben. Ergänzende Materialien werden im Internet unter www.java-hamster-modell.de bereitgestellt.[2]

[1] Text aus dem Benutzerhandbuch des Hamster-Simulators übernommen
[2] Text aus dem Benutzerhandbuch des Hamster-Simulators übernommen

## 1.2 Der Hamster Simulator

Auf der Website www.java-hamster-modell.de steht auch der „Hamster-Simulator“ kostenlos zur Verfügung; ein Programm, mit dem Hamster-Programme erstellt und ausgeführt werden können. Neben den beiden Büchern kommt dem Hamster- -Simulator dabei eine ganz wichtige Bedeutung zu, denn Programmieren lernt man nicht durch lesen. Man muss üben, üben, üben. Und genau dazu dient der Simulator.[3]

Das Hamster-Simulator-Plugin für das „Eclipse SDK“ ist eine Weiterentwicklung des „Hamster-Simulators“ (Version 2.2), wobei diese Version nur in Verbindung mit dem Open Source Framework „Eclipse SDK“ lauffähig ist. Dabei wird diese Version des „Hamster Simulators“ als Plugin in das „Eclipse SDK“ eingebunden (sh. 2).
Dieses Plugin des „Hamster Simulators“ wurde aus verschiedenen Gründen entwickelt. Zum einen wird der „Hamster-Simulator“ durch die Verwendung des „Eclipse SDK“ um zahlreiche Funktionen erweitert und zum anderen wird die Benutzung insgesamt komfortabler. Dies betrifft insbesondere Probleme, die bei der Installation von früheren Versionen aufgetreten sind, aber auch die Dateiverwaltung oder das Schreiben von Hamster-Programmen. Eine detaillierte Übersicht wird im nächsten Abschnitt 1.3 gegeben. Ein weiterer Grund für die Entwicklung eines Hamster-Simulator-Plugins besteht in der Verwendung des „Eclipse SDK“ selbst, da es sich dabei um eine vollwertige Java-Entwicklungsumgebung handelt. Das bedeutet, das man mit Eclipse „reine“ Java-Programme schreiben kann, was mit dem „Hamster-Simulator“ (Version 2.2) nicht möglich ist. Eine Umgewöhnung in eine andere Programmierumgebung ist beim Umstieg von Hamster-Programmen auf Java-Programme somit nicht mehr nötig. Sollte man sich dennoch entscheiden, auf eine andere Programmierumgebung, wie beispielsweise „Netbeans“ oder „Borland JBuilder“, zu wechseln, so wird der Umstieg leichter fallen, denn viele bekannte Funktionen des „Eclipse SDK“ lassen sich in gleicher oder ähnlicher Form in diesen Programmierumgebungen wiederfinden.

Dieses Handbuch „Hamster-Simulator-Plugin für das Eclipse SDK“ ist als eine Ergänzung zum Benutzerhanbuch des „Hamster-Simulators“ (Version 2.2) zu sehen und beschreibt die Benutzung des Hamster-Simulator-Plugins. Die wesentlichen Konzepte ändern sich nicht, jedoch wird der Funktionsumfang des Hamster-Simulators durch die Integration in Eclipse stark erweitert, so das empfohlen wird, sich vorher mit den Grundlagen des „Hamster Simulators“ vertraut zu machen. Alle wesentlichen Konzepte zum Umgang mit dem Java-Hamster-Plugin werden in diesem Handbuch vermittelt, da dieses aber zu großen Teilen auf den Funktionen des „Eclipse SDK“ aufbaut und durch dessen enormen Funktionsumfang nicht alle Konzepte bis ins Detail hier erläutert werden können, sei bei Fragen zusätzlich auf die Hilfe des „Eclipse SDK“ verwiesen.

## 1.3 Änderungen gegenüber des Hamster-Simulators

Mit Hilfe des Hamster-Simulator-Plugins wird der „Hamster Simulator“ in die Programmierumgebung „Eclipse SDK“ integriert. Dadurch werden im „Hamster Simulator“ alle Funktionen die Eclipse bietet nutzbar. Da jedoch nicht alle Funktionen im Alltag eines Hamsterprogrammierers benötigt werden, werden hier nur die wesentlichen Funktionen kurz erläutert, die es

[3] Text aus dem Benutzerhanbuch des Hamster-Simulators übernommen

im bisherigen „Hamster Simulator“ so nicht gab.

- **Variables Fensterkonzept**
  Anders als Im „Hamster Simulator“ (Version 2.2) sind die Komponenten (Editor, Dateiverwaltung, Debugger, Simulation, Toolbars) nicht fest miteinander verbunden, sondern sind in jeweils eigenen Fenstern (Views bzw. Editoren) untergebracht. Diese Fenster können maximiert, minimiert, geschlossen und in einem beliebigen Layout angeordnet werden. Das anfängliche Layout der einzelnen Komponenten ist in der Hamster-Perspektive abgespeichert, welches sich am Layout des bisherigen „Hamster Simulators“ orientiert. Daneben gibt es eine Hamster-Debug-Perspektive, welches ein bestimmtes Layout für das Debuggen von Hamster-Programmen vorgibt. Der Benutzer kann das Layout jederzeit ändern und eigene Perspektiven definieren.
- **Workspace**
  Teil des Eclipse-Konzeptes ist die Verwendung eines *Workspaces*, zu deutsch Arbeitsbereich, in welchem alle Dateien die man anlegt, nach Projekten gruppiert, aufbewahrt werden. Das Workspace-Verzeichnis wird beim ersten Start des „Eclipse SDK“ angelegt und bei zukünftigen Starts verwendet, sofern man dieses als Standard-Workspace-Verzeichnis gewählt hat. Ansonsten wird bei jedem Start nach dem Workspace-Verzeichnis gefragt.
- **Hamster-Projekte**
  Da es in Eclipse möglich ist neben Hamster-Programmen auch Java-Programme oder Programme anderer Programmiersprachen zu schreiben, ist es erforderlich Hamster-Projekte innerhalb des Workspaces anzulegen. Hamster-Projekte werden direkt als Verzeichnis in der Dateiverwaltung angelegt und angezeigt und in ihnen können neue Ordner und Hamster-Programme erzeugt werden. Durch das Speichern von Hamster-Programmen in Hamster-Projekten entfällt zum einen das explizite Setzen eines *Classpath*, wie es im bisherigen „Hamster Simulator“ der Fall war. Zum anderen entfällt das explizite *Kompilieren* von Hamster-Programmen. Dies wird automatisch beim Speichern des jeweiligen Hamster-Programms durchgeführt.
- **Dateiendungen**
  Im „Hamster Simulator“ (Version 2.2) haben Hamsterdateien die Dateiendung „*.ham“. Im Hamster-Simulator-Plugin werden die Dateiendungen nach der Art des Hamster-Programms unterschieden, so dass imperative Hamster-Programme die Dateiendung „*.iham“, objekt-orientierte Hamster-Programme die Dateiendung „*.oham“ und Hamster-Klassen die Dateiendung „*.cham“ haben. Hamsterdateien mit der Dateiendung „*.ham“ werden vom Hamster-Simulator-Plugin unterstützt und können ganz normal im Hamster-Editor geöffnet werden. Diese werden dann in das entsprechende Format des Hamster-Simulator-Plugins umgewandelt und die „*.ham“-Datei wird gelöscht. Um Dateien des Hamster-Simulator-Plugins auch im Hamster-Simulator verwenden zu können, kann man diese in „*.ham“-Dateien zurückwandeln (sh. 4.8).
- **Unterstützung beim Schreiben von Hamster-Programmen**
  Das Schreiben von Hamster-Programmen erfolgt im Hamster-Simulator-Plugin im Hamster-Editor. Dieser verfügt über eine Reihe zusätzlicher Funktionen die es im Editor des bisherigen „Hamster Simulators“ nicht gab:
  - Automatische Ergänzung von Hamster-Befehlen (Code-Assistenz)
  - Syntax-Überprüfung zur Laufzeit
  - Hervorheben von markierten Bezeichnern

  - Code Formatierung
  - Anzeigen von Kommentaren (Javadoc) wenn man auf einem Hamsterbefehl mit der Maus verweilt
  - Suche von Referenzen oder Deklarationen von markierten Bezeichnern
  - Automatisches generieren der geerbten Methoden die man in einer abgeleiteten Klasse überschreiben möchte
  - Automatisches generieren von *Getter*- und *Setter*-Methoden
  - Verwendung von Templates zur Erzeugung von häufig genutzten Programmierkonzepten (z.B. Schleifen)

  Alle anderen Funktionen wie „Schritt zurück“ (undo), „Schritt vorwärts“ (redo), „Kopieren“, „Ausschneiden“, „Einfügen“, „Markieren, „Alles markieren“, „Speichern“ und „Speichern unter“ sowie „Hervorheben von Schlüsselwörtern“ werden weiterhin unterstützt.

- **Unterstützung beim Debuggen von Hamster-Programmen**
  Neben dem aus dem bisherigen „Hamster Simulator“ bekannten Debug-Fenster und Variablen-Fenster, kann beim Debuggen von Hamster-Programmen der von Eclipse bereitgestellte „Expressions View“ verwendet werden. Dieser erlaubt das Angeben von beliebigen Ausdrücken, die bei angehaltener Programmausführung ausgewertet werden können und somit neben den jeweiligen Variablenwerten nützliche Informationen beim Debuggen liefern. Des Weiteren wird von Hamster-Programmen, die im „debug“-Modus ausgeführt werden die „Hot Swap“- bzw. „Hot Code Replace“-Funktion unterstützt. Das bedeutet, das man Änderungen am Source-Code vornehmen kann, während das Hamster-Programm im „debug“-Modus angehalten wurde. Die Ausführung wird dann mit dem veränderten Code fortgesetzt ohne das ein Neustart des Hamster-Programms notwendig ist.
- **Die Simulation**
  Die Simulationskomponente des „Hamster Simulators“ (Version 2.2) wird im Hamster-Simulator-Plugin durch den Territorium-Editor ersetzt. Dieser visualisiert das Territorium und annimiert die Simulation. Anderes als in der bisherigen Simulationskomponente befinden sich die Buttons für die verschiedenen Funktionen und Werkzeuge, wie beispielsweise „Mauern setzen“ oder „Körner im Maul des Hamsters festlegen“ in der globalen Toolbar und sind nur sichtbar, wenn der Territorium-Editor sichtbar und aktiviert ist.
- **Neue Hamster-Befehle**
  In der Simulationskomponente ist ein neues „Log-Panel“ (unterhalb des Simulationsfensters) zu finden, in welchem die Ausgaben der neuen Hamster-Befehle visualisiert werden. Dabei handelt es sich um den Befehl `print("text")` und `println("text")` mit welchen beliebige Textausgaben während der Ausführung des Hamster-Programms geschrieben werden können. Dabei unterscheidet sich `println("text")` von `print("text")` durch seinen automatischen Zeilenumbruch. Mit Hilfe von `"\n"` im Text können zusätzlich explizite Zeilenumbrüche erzwungen werden. Bsp.: `print("text  \n text")`. Durch die Verwendung von `"\t"` im Text können zusätzlich „Tabulatoren“ eingebaut werden. Bsp.: `print("text \t text")`.

Das Hamster-Simulator-Plugin bietet jedoch nicht nur Vorteile gegenüber dem „Hamster Simulator“ in der Version 2.2, sondern hat auch einige Nachteile. Dazu gehört, dass durch den gestiegenen Funktionsumfang die intuitive Bedienbarkeit des „Hamster Simulators“ vermindert wird. Des Weiteren ist das Hamster-Simulator-Plugin nicht in der Lage mit der Entwicklungsumgebung „BlueJ“ zusammenzuarbeiten.

# 2 Installation und Starten des Hamster-Simulator-Plugins

In diesem Kapitel wird nun zunächst darauf eingegangen, welche Software bereits installiert sein muss, um das Hamster-Simulator-Plugin zu installieren. Anschließend wird auf die Installation selbst eingegangen und welche verschiedenen Möglichkeiten es gibt und abschließend wird erläutert, wie man das Hamster-Simulator-Plugin startet.

## 2.1 Vorraussetzungen

Um das Hamster-Simulator-Plugin installieren zu können muss bereits bestimmte Software im System installiert worden sein. Dies ist zu einen die Java-Laufzeit-Umgebung in der Version 5.0 oder höher und das „Eclipse SDK“ in der Version 3.1.2 oder höher.

Die Java Laufzeitumgebung, in der Version „ J2SE(TM) Development Kit 5.0“ kann von folgenden Webseite bezogen werden: http://java.sun.com/javase/downloads/index.jsp

Das „Eclipse SDK“ kann von der folgenden Seite bezogen werden: http://www.eclipse.org/downloads/

## 2.2 Installation des Hamster-Simluator-Plugins

Die Installation des Java-Hamster-Plugins kann auf zwei verschiedene Arten erfolgen. Zum einen kann dies mit Hilfe des „Update Managers“ von Eclipse erfolgen und zum anderen durch das explizite entpacken der Plugin-Archivdatei in das Plugin-Verzeichnis von Eclipse. Im folgenden werden beide Möglichkeiten mit ihren Vor- und Nachteilen erläutert:

- **Installation mit Hilfe des „Update Managers“ von Eclipse**

  1. Starten Sie Eclipse und achten Sie dabei auf die eingeblendete Versionsnummer!
  2. Wählen Sie im Menü „Help → Software Updates → Find and Install. . . “.
  3. Es öffnet sich das Fenster „Feature Updates“, wo Sie den Radio Button „Search for new features to install“ selektieren und anschließend „Next“-Button betätigen.
  4. Es öffnet sich nun das Fenster „Update sites to visit“. Dort wählen Sie den Button „New Remote Site“ und geben in dem folgenden Dialog für Name-Feld „HamsterUpdateSite“ ein. Im URL-Feld geben Sie „http://www-is.informatik.uni-oldenburg.de/~dibo/hamster/HamstersimulatorPlugin/“ ein, was der Adresse der Hamster-Update-Site entspricht. Anschließend wählen Sie den „OK“-Button des Dialog Felds um dieses zu schließen. Im Fenster „Update sites to visit“ selektieren Sie nun nur „HamsterUpdateSite“ und wählen anschließend den „Finish“-Button.

5. Es öffnet sich wiederum ein Fenster mit den „Search Results“. In diesem steht unter „Select the features to install“ HamsterUpdateSite. Klappen Sie dieses auf, indem Sie auf das „Plus“ Symbol klicken. Als nächstes wählen Sie „HamstersimulatorPlugin_X“, wobei X für ihre Eclipse Version steht. Sollte ihre Eclipse Version nicht aufgeführt sein, so wählen Sie bitte die Eclipse Version, welche ihrer am nächsten kommt. Dabei müssen Sie darauf achten, das Sie abrunden und nicht aufrunden. Das bedeutet, wenn Sie beispielsweise Version 3.2.3 oder 3.3.5 verwenden und 3.2 und 3.4 zur Auswahl stehen, wählen Sie 3.2 aus. Fahren Sie fort, indem Sie den „Next“-Button drücken.
6. In dem nun folgenden Fenster wird ihnen die Lizenz des Hamster-Simulator-Plugins angezeigt. Lesen Sie diese sorgfältig durch und entscheiden Sie sich dann, ob Sie diese akzeptieren wollen. Sie können nur fortfahren, wenn Sie die Lizenz akzeptieren. Durch den „Next“-Button gelangen Sie zum Fenster „Installation“.
7. Im Fenster „Installation“ können Sie wählen, wo das Plugin installiert werden soll. Es wird dringend empfohlen das angegebene Verzeichnis, welches in der Regel ihrem Eclipse-Installationsverzeichnis entspricht beizubehalten, da es sonst bei der Ausführung des Plugins zu Problemen kommen kann. Durch Betätigen des „Finish“-Buttons schließen Sie die Installation ab.
8. Das Plugin wird nun heruntergeladen und es erscheint eine Meldung das Sie ein „unsigned feature“ installieren. Dies ist richtig. Fahren Sie durch Betätigen „install“-Buttons mit der Installation fort.
9. Zum Schluss erscheint eine Meldung, das die „Workbench“ neu gestartet werden sollte, um die Änderungen am „Eclipse SDK“ zu übernehmen. Bestätigen Sie diesen Dialog mit „yes“. Eclipse wird nun neu gestartet und Sie können das Hamster-Simulator-Plugin verwenden.

- **Installation durch Entpacken der Archivdatei**
  1. Als erstes müssen Sie ihre Eclipse-Version ermitteln. Dies Können Sie tun indem Sie Eclipse starten und auf die eingeblendete Versions-Nummer im Startbildschirm achten.
  2. Als nächstes Laden Sie sich das Hamster-Simulator-Plugin von der Seite http://www.java-hamster-modell.de/HamstersimulatorPlugin/plugins herunter. Die Plugin Archivdatei beginnt mit org.uni_ol.hamstersimulator_x.x.y.jar, wobei die x.x der Versionsnummer von Eclipse entspricht und y der Versionsnummer des Plugins. Achten Sie beim Download darauf, das die Eclipse-Versionsnummer der Plugin-Archivdatei ihrer Eclipse-Version entspricht.
  3. Entpacken Sie die „*.jar“-Archivdatei in den Plugin-Ordner ihres Eclipse, beispielsweise mit dem Programm „Winrar“.
  4. Starten Sie Eclipse neu, sofern Eclipse noch lief.

## 2.3 Starten des Hamster-Simluator-Plugins

Um das Hamster-Simulator-Plugin zu Starten muss das „Java JDK“, „Eclipse SDK“ und das Hamster-Simulator-Plugin wie oben beschrieben installiert worden sein. Das Hamster-Simulator-Plugin wird nun automatisch beim Start des „Eclipse SDK“ geladen und dessen

Funktionen sind verfügbar.

# 3 Ihr erstes Hamster-Programm

Haben Sie das „Eclipse SDK“ mit dem Hamster-Simulator-Plugin gestartet, so werden Sie sich zunächst in der Java-Perspektive oder in der Resources-Perspektive befinden. In diesen Perspektiven sind nicht nur die Funktionen des Hamster-Simulators, sondern auch eine Vielzahl anderer Funktionen des „Eclipse SDK“, verfügbar. Um diesen Funktionsumfang zu minimieren, gibt es eine spezielle Hamster-Perspektive und eine Hamster-Debug-Perspektive, welche den Schwerpunkt auf Funktionen des „Hamster Simulators“ legen. Daher ist es ratsam in die Hamster-Perspektive zu wechseln. Dies erreichen Sie, indem Sie im Menü „Window“ auf „Open Perspektive → Other“ klicken und in dem nun erscheinenden Dialogfeld „Hamster Perspektive“ wählen.

Im folgenden wird Ihnen nun gezeigt, wie Sie Ihr erstes Hamster-Programm im Hamster-Simulator-Plugin schreiben können. Dabei werden ähnlich wie im bisherigen Hamster-Simulator verschiedene Stationen durchlaufen. Dazu gehören:

- Erstellen eines Hamster-Projekts
- Gestalten eines Hamster-Territoriums
- Eingeben eines Hamster-Programms
- Ausführen eines Hamster-Programms
- Debuggen eines Hamster-Programms

Dabei werden im folgenden die einzelnen Szenarien kurz skizziert, jedoch nicht alle Funktionen des Hamster-Simulator-Plugins im Detail erklärt. Dies erfolgt in dem folgenden Kapitel „Bedienung des Hamster-Simulator-Plugins“ (sh. 4).

## 3.1 Erstellen eines Hamster-Projekts

Anders als im bisherigen „Hamster Simulator“ müssen nun Hamster-Projekte im Workspace erstellt werden. Dadurch wird es möglich, Hamster-Projekte klar von anderen Projekten, wie beispielsweise Java-Projekten, abzugrenzen und spezifische Einstellungen für Hamster-Projekte vorzunehmen, wie z.B. das Setzen des „Classpath“.

Um nun ein Hamster-Projekt zu Erstellen wählen Sie im Menü „File“ den Eintrag „New → Hamster Project“ aus. Es öffnet sich der „Hamster Projekt Wizard“ welcher die Erzeugung des Hamster-Projekts vornimmt. Sie müssen an dieser Stelle den Namen des Hamster-Projekts eingeben und den „Finish“-Button betätigen. Es wird nun ein neues Hamster-Projekt mit dem gewählten Namen im Workspace-Verzeichnis angelegt und erscheint im „Navigator-View“ (sh. 4.1.3), welcher sich am linken Rand des Bildschirms in der Hamster-Perspektive befindet.

Sollten Sie im „Problems View„ (sh. 4.1.7) nun eine Fehlermeldung der Art: „Project ’Projektname’ is missing required library: ’library-Name’“, so wurde der „Build Path“ bzw. Classpath

in ihrem System nicht richtig gesetzt. Bitte befolgenden Sie in diesem Fall die Anweisungen im Abschnitt 4.2.4.

## 3.2 Gestalten eines Hamster-Territoriums

Im bisherigen „Hamster Simulator“ wurde das Territorium im Simulations-Fenster des „Hamster Simulators“ angezeigt. Dies ist im Hamster-Simulator-Plugin nicht der Fall. Das Territorium wird im Territorium-Editor (sh. 4.1.4) angezeigt, welcher sich in der „Editor-Area“ befindet.

Um ein Territorium zu Erstellen bzw. Anzuzeigen gibt es im Hamster-Simulator-Plugin verschiedene Möglichkeiten:

- Die erste Möglichkeit ist das Erstellen eines Territoriums mit Hilfe des „Hamster Terrain Wizards“. Diesen öffnen Sie durch wählen des Eintrags „New → Hamster Terrain“ im Menü „File“. Im „Hamster Terrain Wizard“ muss das Hamster-Projekt, der Dateiname sowie die Größe des Territoriums angegeben werden. Durch betätigen des „Finish“-Buttons wird ein neues Territorium im gewählten Hamster-Projekt erzeugt und im Territorium-Editor angezeigt.
- Die zweite Möglichkeit ist die Verwendung des Standard-Territoriums, welches im Territorium-Editor geöffnet wird, sobald eine neue Hamsterdatei erstellt oder eine bestehende Hamsterdatei geöffnet wird. Das Standard-Territorium zeigt initial ein Territorium der Größe 14x10 Kacheln. Allerdings verhält sich das Standard-Territorium so, dass immer das zuletzt bearbeitete Territorium angezeigt wird. Das bedeutet, wurde ein anderes Territorium erstellt und bearbeitet, so wird beim Aufruf des Standard-Territoriums dieses angezeigt.
- Die dritte Möglichkeit ist die Verwendung eines bereits bestehenden Territoriums. Diese werden im „Navigator-View“ angezeigt (Mauer-Symbol) und lassen sich durch Doppelklick (Windows) oder über dessen Kontextmenü (rechte Maustaste) und den Eintrag „open“ öffnen.

Haben Sie ein Territorium im Territorium-Editor geöffnet, so erscheint in der globalen Toolbar (sh.4.1.1) eine Reihe von Buttons zum Bearbeiten des Territoriums. Diese sind bereits aus bisherigen „Hamster Simulator“ bekannt und erlauben unter anderem das:

- Heinein- und Hinaus-Zoomen - Vergrößert und verkleinert die dargestellte Größe des Territoriums
- Setzen des Standard-Hamsters - Wählen Sie dieses Werkzeug aus, um den Standard-Hamster im Territorium auf einer anderen Kachel zu positionieren. Dazu klicken Sie mit einer Maustaste die Kachel an, auf der der Standard-Hamster sitzen soll. Die Kachel darf nicht durch Mauern versperrt sein.
- Setzen der Körner im Maul des Hamsters - Mit diesem Werkzeug können Sie die Anzahl der Körner im Maul des Hamsters verändern. Dabei ist die initiale Körnermenge gleich null (0). Negative Werte sind nicht zulässig.
- Drehen des Standard-Hamsters um 90° nach links - Dieser Button dient dem Drehen des Standard-Hamsters um 90° nach links.
- Setzen von Wänden - Mit diesem Werkzeug können Wände im Territorium positioniert

werden. Dazu klicken Sie mit einer Maustaste auf das Feld, auf dem eine Mauer erscheinen soll. Sollen mehrere Mauern auf einmal gesetzt werden, so wählen Sie mit einer Maustaste die Startkachel und halten Sie die Maustaste gedrückt, während Sie mit der Maus zur Zielkachel gehen. Lassen Sie dort die Maustaste los. Auf allen überdeckten Kacheln wird eine Mauer gesetzt. Es wird keine Mauer gesetzt, wenn die Kachel durch den Standard-Hamster verdeckt ist.

- Setzen von Körnern - Das Setzen von Körnern funktioniert analog zum Setzen von Mauern, jedoch erscheint nach loslassen der Maustaste eine Dialogbox, in welcher Sie die Anzahl der Körner auf der Kachel festlegen müssen. Negative Werte sind dabei nicht erlaubt.
- Löschen von Wänden und Körnern - Um Wände und Körner auf den Kacheln zu löschen selektieren Sie das Lösch-Werkzeug und klicken auf die Kachel dessen Inhalt gelöscht werden soll. Auch das Löschen mehrerer Kacheln ist möglich. Dazu selektieren Sie eine Startkachel und halten die Maustaste gedrückt und gehen mit der Maus zur Zielkachel. Dort lassen Sie die Maustaste los. Alle überdeckten Felder werden gelöscht.
- Zurücksetzen des Territorium nach einer Simulation - Dieser Button dient dem Zurücksetzen des Territoriums in den Zustand, wie er vor dem Starten der Simulation gewesen ist.
- Erhöhen der Simulationsgeschwindigkeit - Mit diesem Button kann die Simulationsgeschwindigkeit erhöht werden. Wird der Button deaktiviert so ist keine weitere Erhöhung der Geschwindigkeit möglich.
- Verringern der Simulationsgeschwindigkeit - Mit diesem Button kann die Simulationsgeschwindigkeit verringert werden. Wird der Button deaktiviert so ist keine weitere Verringerung der Geschwindigkeit möglich.
- Löschen der Logausgaben nach einer Simulation - Beim Ablaufen von Simulationen werden in den Logfenstern Ausgaben des Hamster-Programms visualisiert. Diese können durch drücken dieses Buttons gelöscht werden.

Alle Buttons verfügen über eine „Tooltip“-Text, welcher bei verweilen mit der Maus über dem Button angezeigt wird und Informationen über die Funktion des Buttons liefert. Des Weiteren bedingen sich die Werkzeuge gegenseitig, was bedeutet das nur ein Werkzeug zur Zeit aktiviert sein kann. Die Werkzeuge funktionieren auch, während eine Simulation abläuft.

Mit diesen „Werkzeugen“ und Funktionen können Sie das Territorium nun so modellieren, dass spezifische Aufgaben damit gelöst werden können. Wenn Sie das Territorium zu einem späteren Zeitpunkt wieder verwenden wollen, so müssen Sie dieses explizit abspeichern. Dazu wählen Sie im Menü „File“ den Eintrag „Save As . . . “. Es erscheint der „Save As“-Dialog in welchem Sie das Hamster-Projekt wählen und den Namen des Territoriums eingeben, unter dem Sie das Territorium speichern wollen. Durch Betätigen des „OK“-Buttons beenden Sie den Vorgang.

Versuchen Sie nun das in 3.1 dargestellte Territorium zu modellieren. Dies wird für das spätere Hamster-Programm benötigt.

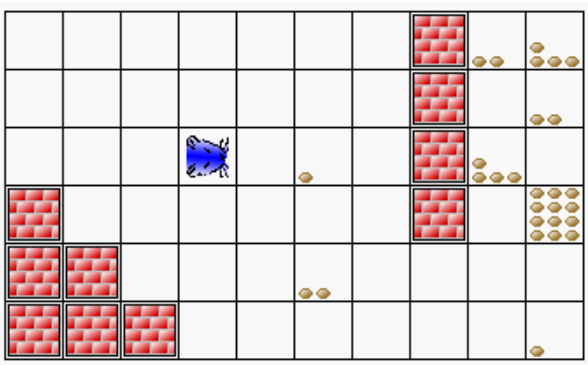

Abbildung 3.1: Hamster-Territorium

## 3.3 Eingeben eines Hamster-Programms

Um ein neues Hamster-Programm einzugeben ist es zunächst erforderlich, dass Sie eine neue Hamsterdatei im Workspace erstellen, in welcher das Hamster-Programm gespeichert wird. Dazu rufen Sie den „Hamster File Wizard“ über das Menü „File“ „New → Hamster File“ auf. In diesem Wizard müssen Sie zunächst das Hamster-Projekt und den Namen der Hamsterdatei angeben. Dabei ist beim Name der Hamsterdatei darauf zu achten, dass der Name ein gültiger Java-Bezeichner ist, was bedeutet, dass der Name nur ein Wort sein darf! Am besten wählen Sie hier „MeinErstesProgramm“. Anschließend müssen Sie die Art der Hamsterdatei wählen. Dabei können imperative Hamsterdateien von objektorientierten Hamsterdateien und Klassen-Hamsterdateien unterschieden werden. Dabei speichern imperative Hamsterdateien imperative Hamster-Programme, objektorientierte Hamsterdateien objektorientierte Hamster-Programme und Klassen-Hamsterdateien Hamster-Klassen ab.

Im Folgenden soll nun ein imperatives Hamsterprogramm eingegeben werden. Dazu wählen Sie „imperativ“ im „Hamster File Wizard“ aus und betätigen den „Finish“-Button zum Beenden des Wizards. Es wird eine neue imperative Hamsterdatei im angegebenen Hamster-Projekt erzeugt und diese Hamsterdatei im Hamster-Editor geöffnet. Der Hamster-Editor ist der zentrale Editor zum Schreiben von Hamsterdateien und bietet eine Reihe von Funktionen, die beim Schreiben von Hamster-Programmen nützlich sind. Auf diese wird im Kapitel Hamster-Editor (sh. 4.1.5) näher eingegangen. Hamster-Programme können natürlich auch mit jedem anderen in Eclipse verfügbaren Editor geschrieben werden, jedoch fallen dann die speziellen Unterstützungfunktionen weg.

Im Hamster-Editor steht nun:

```
//Die Import-Anweisungen - nicht editieren
import de.hamster.model.client.Hamster;
import de.hamster.model.client.Territorium;
import de.hamster.model.client.HamsterInitialisierungsException;
import de.hamster.model.client.HamsterNichtInitialisiertException;
import de.hamster.model.client.KachelLeerException;
import de.hamster.model.client.MauerDaException;
import de.hamster.model.client.MaulLeerException;
```

```
import de.hamster.model.client.HamsterException;
import de.hamster.model.client.HamsterProgram;
public class MeinErstesProgramm extends Hamster implements HamsterProgram {
      public void main(){

      }
}
```

Die „import“-Anweisungen werden im Hamster-Editor in der Regel minimiert dargestellt und müssen von Ihnen nicht weiter beachtet werden. Allerdings dürfen Sie diese Anweisungen auch nicht löschen oder editieren! Auch der Klassenrumpf (public class 'gewählter_Name' extends ...) darf nicht geändert werden und muss von ihnen nicht weiter beachtet werden.

Das eigentliche Hamster-Programm geben Sie zwischen `public void main(){` und `}` ein. Im Folgenden sollen Sie ein Programm schreiben, dass den Standard-Hamster befähigt, zwei der Körner in seiner Nähe im Territorium 3.1 aufzunehmen. Dazu geben Sie folgendes ein:

```
public class MeinErstesProgramm extends Hamster implements HamsterProgram {
  public void main() {
    vor();
    vor();
    nimm();
    rechtsUm();
    vor();
    vor();
    nimm();
  }

  void rechtsUm() {
    linksUm();
    linksUm();
    linksUm();
  }
}
```

Um die Datei abzuspeichern klicken Sie das Speichern-Symbol (Diskette) in der globalen Toolbar oder wählen Sie im Menü „File“ den Eintrag „Save“. Durch das Speichern wird die Datei kompiliert und mögliche Fehler werden im „Problems View“ angezeigt. Allerdings verfügt der Hamster-Editor über die Möglichkeit syntaktische Fehler auch schon bei der Eingabe des Hamster-Programms zu erkennen und Ihnen Vorschläge zu unterbreiten. Um die Datei unter einem anderen Namen zu speichern kann im Menü „File“ der Eintrag „Save As“ gewählt werden, welcher den „Save As“-Dialog öffnet. In diesem müssen Sie das Projekt sowie den Namen der Hamsterdatei mit Dateiendung eingeben.

Drucken können Sie das Hamster-Programm, indem Sie in der globalen Toolbar das „Drucker-Symbol“ betätigen. Alternativ können Sie auch im Menü „File“ den Eintrag „Print“.

Nachdem Sie das Hamster-Programm nun eingegeben haben, prüfen Sie bitte, dass keine syntaktischen Fehler mehr vorhanden sind. Syntaktische Fehler können Sie an rot unterschlän-

gelten Ausdrücken im Programmcode erkennen, an einem roten Kreis mit weißem Kreuz (Marker) am linken Rand des Editors oder an einer Error-Meldung im „Problems View“. Sind noch Fehler vorhanden, so lesen Sie sich den Tooltip durch, der erscheint wenn Sie über dem Fehler/Marker mit der Maus verweilen. In einigen Fehlerfällen kann Eclipse auch direkt den Fehler für Sie beheben. In diesem Fall hat der Marker die Form einer Glühlampe mit einem Roten Kreis und einem weißen Kreuz. Sie können dann mit der linken Maustaste auf diesen Marker klicken, um die „Quick Fix“ Hilfe zu erhalten.

## 3.4 Ausführen eines Hamster-Programms

Nachdem Sie das Programm nun eingegeben haben und alle syntaktischen Fehler beseitigt sind können Sie das Hamsterprogramm starten. Dazu gehen Sie ins Kontexmenü (rechte Maustaste) des Editors mit dem Hamster-Programm und wählen dort den Eintrag „Run As → Hamsterprogramm“. Der Territorium-Editor wird automatisch in den Vordergrund geholt bzw. neu geöffnet wenn er geschlossen worden ist. Das Hamster-Programm läuft nun ab. Das Hamster-Programm stoppt automatisch, wenn der Programm-Code vollständig abgearbeitet ist. Als Benutzer haben Sie die Möglichkeit das Hamster-Programm auch explizit zu stoppen, indem Sie in der globalen Toolbar den „Stopp“-Button (rotes Quadrat) drücken. Alternativ finden Sie den „Stopp“-Button auch im „Debug View“ und im Menü „Hamstersimulator“. Das Pausieren und Fortführen von Hamster-Programmen ist, anders als im bisherigen „Hamster Simulator“, im „run“-Modus nicht mehr möglich, sondern nur im „debug“-Modus. Dies liegt daran, dass „spätere“ Java-Programme auch nur im „debug“-Modus angehalten werden können und somit das Verhalten einheitlich ist.

Während der Ausführung des Hamster-Programms werden Logausgaben in den „Log-Panel“(rechts vom Simulationsfenster) im Territorium-Editor angezeigt. Dabei handelt es sich um die Hamster-Befehle, die von den Hamstern ausgeführt werden, aber auch um Exceptions oder Informationen über die aktuelle Ausführungsgeschwindigkeit.

Der „Log-Panel“ unterhalb des Simulationsfensters wurde im Hamster-Simulator-Plugin neu eingeführt und zeigt die Ausgaben, die mittels der neuen Hamster-Befehle `print("text")` und `println("text")` geschrieben werden (sh. 1.3).

Mit Hilfe der Buttons „Simulationsgeschwindigkeit erhöhen“ und „Simulationsgeschwindigkeit verringern“ in der globalen Toolbar („Pfeil hoch“- und „Pfeil runter“-Symbol) lässt sich auch während der Ausführung des Hamster-Programms dessen Simulationsgeschwindigkeit beeinflussen. Mit dem „Zurücksetzen“-Button lässt sich das Territorium nach der Simulation zurücksetzen.

Nach erfolgreicher Ausführung ihres Hamster-Programms sollte der Standard-Hamster nun das in Abbildung 3.1 zu sehende Korn auf der Position (6/3) und eins der Körner auf Position (6/5) gefressen haben. Betätigen Sie den Zurücksetzen-Button, um die Ausgangsposition wiederherzustellen.

## 3.5 Debuggen eines Hamster-Programms

Das Debuggen eines Hamster-Programms läuft prinzipiell ähnlich zum Ausführen eines Hamster-Programms ab, jedoch werden zusätzliche Funktionen bereitgestellt, die Ihnen helfen ihr Hamster-Programm auf semantische Korrektheit zu überprüfen. Wechseln Sie in die Hamster-Debug-Perspektive, indem Sie im Menü „Window“ den Eintrag „Open Perspective → Hamster Debug Perspektive“ wählen. Das Layout der Komponenten hat sich nun geändert, so dass Sie oben links den „Debug View“ und oben rechts den „Variables View“ sowie den „Breakpoints View“ sehen. Darunter befinden sich der Hamster-Editor, sowie der Territorium-Editor, die nebeneinander angeordnet sind. Sollten Sie den Territorium-Editor nicht sehen, so selektieren Sie den Hamster-Editor und wählen in der globalen Toolbar das „Mauer“-Symbol, welches automatisch das Standard-Territorium öffnet.

Um ein Hamster-Programm im „debug“-Modus zu starten wählen Sie im Kontextmenü des Hamster-Editors den Eintrag „Debug As → Hamsterprogramm“. Die Ausführung des Hamster-Programms wird nun gestartet und verhält sich wie die Ausführung im „run“-Modus. Sie haben aber nun zusätzlich die Möglichkeit, über das Pause- und Fortfahren-Symbol in der globalen Toolbar, das Hamster-Programm zu Pausieren und Fortzufahren .

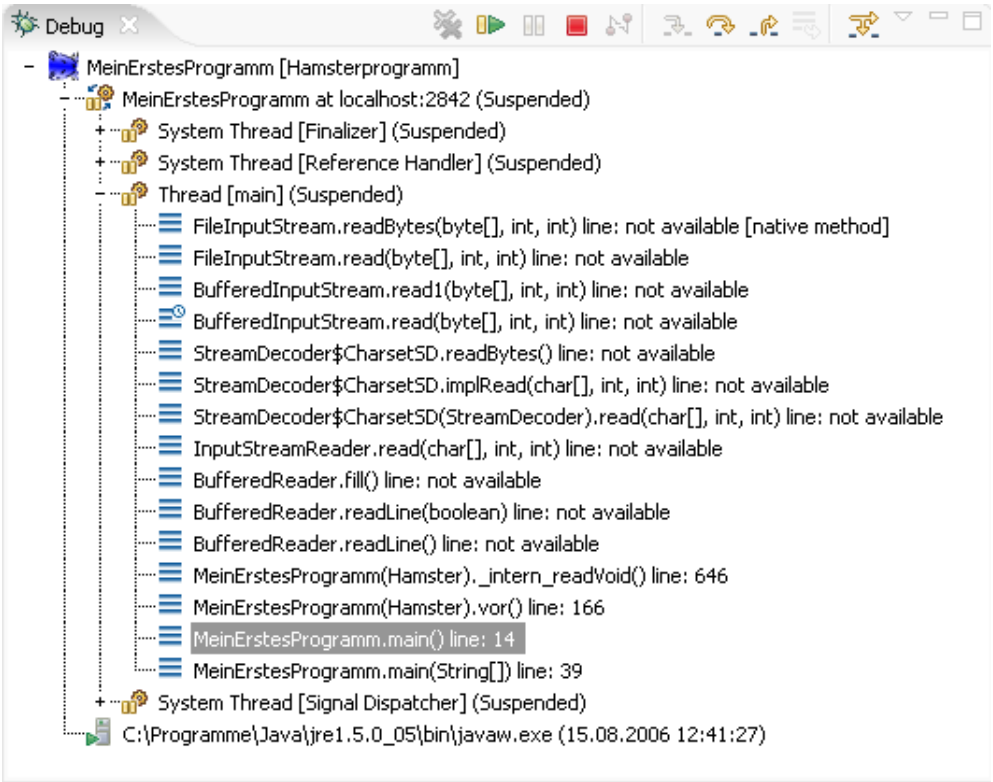
Debug
MeinErstesProgramm [Hamsterprogramm]
MeinErstesProgramm at localhost:2842 (Suspended)
System Thread [Finalizer] (Suspended)
System Thread [Reference Handler] (Suspended)
Thread [main] (Suspended)
FileInputStream.readBytes(byte[], int, int) line: not available [native method]
FileInputStream.read(byte[], int, int) line: not available
BufferedInputStream.read1(byte[], int, int) line: not available
BufferedInputStream.read(byte[], int, int) line: not available
StreamDecoder$CharsetSD.readBytes() line: not available
StreamDecoder$CharsetSD.implRead(char[], int, int) line: not available
StreamDecoder$CharsetSD(StreamDecoder).read(char[], int, int) line: not available
InputStreamReader.read(char[], int, int) line: not available
BufferedReader.fill() line: not available
BufferedReader.readLine(boolean) line: not available
BufferedReader.readLine() line: not available
MeinErstesProgramm(Hamster)._intern_readVoid() line: 646
MeinErstesProgramm(Hamster).vor() line: 166
MeinErstesProgramm.main() line: 14
MeinErstesProgramm.main(String[]) line: 39
System Thread [Signal Dispatcher] (Suspended)
C:\Programme\Java\jre1.5.0_05\bin\javaw.exe (15.08.2006 12:41:27)

Abbildung 3.2: Debug View: Callstack

Während der Ausführung im „debug“-Modus werden im „Debug View“ die einzelnen „Threads“ angezeigt. Dabei entspricht der „Thread“ mit der Aufschrift „main“ der Ausführung des Hamster-Programms. Pausiert man nun den „main-Thread“ und klappt die Baumstruktur mit Hilfe des Plus-Symbols auf , so wird der „Callstack“ des Threads sichtbar. Dies ist in

Abbildung 3.2 zu sehen, wobei es sich beim „Callstack“ um die Aufrufhierachie der einzelnen Methoden handelt. Wie man in der Abbildung 3.2 sieht steht im Callstack zuerst die Klasse, zu welcher die aufgerufene Methode gehört, dann die Methode die aufgerufen wurde und anschließend die Zeile, in welcher die Methode sich im Hamster-Editor befindet. Von Interesse ist für Sie nur ihre eigene Hamster-Klasse. Selektieren Sie wie in Abbildung 3.2 zu sehen die oberste Methode in der Aufrufhirarchie, die noch zu ihrer Hamster-Klasse gehört. Eclipse zeigt Ihnen nun im Editor die Zeile, die der Selektion entspricht und die gerade ausgeführt wird. Sie haben nun die Möglichkeit ihr Hamster-Programm schrittweise auszuführen, indem Sie die drei Pfeiltasten der lokalen Toolbar des „Debug Views“ verwenden. Dabei bedeutet die linke Pfeiltaste „Schritt hinein“, die mittlere „Schritt hinüber“ und die rechte „Schritt hinaus“. Mit „Schritt hinein“ können Sie in die der Aufrufherachie nach oben gehen, dass heißt in die aufzurufende Methode hineinspringen. Mit „Schritt hinüber“ können Sie die Methode ausführen ohne in Sie hinein zu springen und mit „Schritt hinaus“ verlassen Sie die aktuelle Methode und gehen zur Methode zurück, welche die Methode in der Sie sich gerade befanden, aufgerufen hat. Das bedeutet Sie gehen in der Aufrufhierarchie runter.

Versuchen Sie nun zunächst die Pfeiltaste „Schritt hinüber“ (Step Over) um zum nächsten Hamster-Befehl ihrer Klasse zu springen. Sind Sie bei dem Aufruf der Methode „rechtsUm()“ angekommen, so können Sie auch die Pfeiltaste „Schritt hinein“ (Step Into) und die Pfeiltaste „Schritt hinaus“ (Step Return) versuchen, da es sich dabei um eine Methode ihres Hamster-Proramms handelt. Für die anderen Befehle machen die Pfeiltasten „Schritt hinein“ und „Schritt hinaus“ noch keinen Sinn, da Sie Methoden aufrufen, die nicht zu ihrem Hamster-Programm gehören. Allerdings werden diese Funktionen bei komplexeren Hamster-Programmen als nützlich erweisen.

Wenn Sie ein Hamster-Programm pausiert haben und schrittweise ausführen können Sie im „Variables View“ die Variablen ihres Hamster-Programms und deren Werte die Sie zu dieser Zeit haben ablesen. Bei unserem einfachen Hamster-Programm werden aber noch keine Variablen verwendet, so dass nur die Variable „this“ angezeigt wird, welche dem Hamster-Programm entspricht.

Anders als im bisherigen Hamster-Simulator wird vom Hamster-Simulator-Plugin das Konzept der „Breakpoints“ unterstützt. Dies sind „Marker“ die man am linken Rand des Hamster-Editors setzen kann und die Eclipse sagen, dass man die Ausführung des Hamster-Programms an dieser Stelle pausieren möchte, um beispielsweise schrittweise fortzufahren. Dies wollen wir im Folgenden nun üben. Dazu brechen Sie die aktuelle Ausführung des Hamster-Programms bitte ab und versetzen das Territorium wieder in die Ausgangssituation. Setzen Sie nun im Hamster-Editor in der Zeile mit dem Hamster-Befehl `rechtsUm()` einen Breakpoint, indem Sie am linken Seitenrand (links von der Zeilennummer) das Kontextmenü (rechte Maustaste) aufrufen und den Eintrag „Toggle Breakpoint“ wählen. Es erscheint nun ein blauer Punkt im linken Rand des Hamster-Editors. Führen Sie nun ihr Hamster-Programm im „debug“-Modus aus. Wie Sie sehen pausiert Eclipse automatisch die Ausführung des Hamster-Programms in der Zeile mit dem Breakpoint. Sie können nun die Ausführung durch den Fortfahren-Button fortsetzen oder auch mit Hilfe der Pfeiltasten das Programm Zeilenweise ausführen. Löschen können Sie den Breakpoint auf die gleiche Art und Weise, wie sie ihn erzeugt haben, indem Sie „Toggle Breakpoint“ im Kontextmenü wählen. Wollen Sie den Breakpoint nur abschalten, so können Sie auch „Disable Breakpoint“ im Kontextmenü wählen. Der „Breakpoint View“ gibt Aufschluss über alle Breakpoints die in sämtlichen Programmen gesetzt wurden. Diese

können dort gelöscht und abgeschaltet werden.

## 3.6 Zusammenfassung

Herzlichen Glückwunsch! Sie haben nun ihr erstes Hamster-Programm mit dem Hamster-Simulator-Plugin für das „Eclipse SDK“ geschrieben, ausgeführt und gedebuggt. Sie haben sich nun die grundlegenden Kenntnisse angeeignet um mit dem Hamster-Simulator-Plugin zu arbeiten. Da dieser „Hamster Simulator“ in Verbindung von Eclipse noch eine Reihe weiterer zusätzlicher und nützlicher Funktionen bietet, können Sie diese entweder nun selbst erkunden oder im nächsten Kapitel nachlesen.

# 4 Bedienung des Hamster-Simulator-Plugins

In diesem Kapitel wird das Hamster-Simulator-Plugin nun auf einer detailierteren Ebene besprochen und weitere nützliche Funktionen aufgezeigt. Dazu werden nun im Folgenden die einzelnen Bestandteile kurz erläutert und anschließend auf die Funktionen des Hamster-Simulator-Plugins im Detail eingegangen.

## 4.1 Die Bestandteile des Hamster-Simulator-Plugins

### 4.1.1 Globale Toolbar

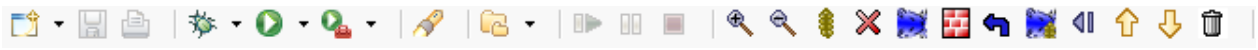

Abbildung 4.1: Globale Toolbar

Wie in Abbildung 4.1 und A.1 zu sehen, beherbergt die globale Toolbar eine Reihe von Actionsets (Gruppe von Buttons), welche je nach Perspektive und aktivierten Fenstern variieren kann. Die globale Toolbar verfügt in der Hamster-Perspektive über den „New“-Button welcher den „Select a Wizard“-Dialog aufruft, in welchem man verschiedene Wizards ausführen kann. Klickt man nicht direkt auf den „New“-Button, sondern auf das „Drop DownMenü des Buttons, so werden einem direkt die drei Hamster-Wizards zur Auswahl angeboten. Rechts neben dem „New“-Button befindet sich der „Speichern“-Button, welcher es erlaubt nicht gespeicherte Dokumente eines selektierten Editors zu speichern. Daneben befindet sich der „Drucken“-Button um den Inhalt eines selektierten Editors zu drucken.

Rechts daneben beginnt das Launch-Actionset, welches die Buttons zum Starten von Programmen im „debug“- und „run“-Modus bereitstellt. Klickt man auf die Buttons, so wird immer das zuletzt ausgeführte Programm im gewählten Modus gestartet. Wählt man hingegen das „Drop Down“-Menü des Buttons so bekommt man die Historie der letzten ausgeführten Programme aufgelistet und kann davon eins wählen. Daneben kann man auch „Run As“ oder „Debug As“ und dann beispielsweise Hamsterprogramm wählen. Dies ist eine „Shortcut“ Funktion und führt die aktuelle Selektion beispielsweise als Hamster-Programm aus. Dabei können mögliche Selektionen ein Hamster-Editor mit einem Hamster-Programm sein oder eine selektierte Hamster-Datei im „Navigator View“. Neben der Shortcut Funktion gibt es im „Drop Down“-Menü dieser Buttons auch den Eintrag „Run...“ und „Debug...“. Dies öffnet das Dialogfeld „Create, manage, and run Configrations“, welches näher im Abschnitt 4.6 erläutert wird.

Rechts neben dem Launch-Actionset befindet sich der „Search“-Button, welcher den „Search“-Dialog öffnet. Dieser erlaubt das eine Datei-spezifische, Java-Spezifische und Plugin-spezifische Suche wobei je nach installierten Plugins im „Eclipse SDK“ auch noch andere Suchtypen möglich sind und in diesem Dialog angezeigt werden. Für weiterführende Hilfe sei auf die

Eclipse Hilfe verwiesen.

Wiederum rechts daneben beginnt der Hamster-Debug-Actionset, welcher die Buttons zum Fortfahren, Pausieren und Stoppen von Hamster-Programmen beherbergt. Dabei sind die Funktionen „Pausieren“ und „Fortfahren“ nur bei der Ausführung von Hamster-Programmen im „debug“-Modus aktiviert und der Stopp-Button ist sowohl im „run“- als auch im „debug“-Modus aktiviert. Dabei sind die Buttons unabhängig von der Perspektive, in der man sich gerade befindet.

Daneben befindet sich schließlich das Hamster-Territorium-Actionset, welches die Werkzeuge zum Modellieren des Territoriums bereitstellt (sh. 3.2). Dieses Actionset ist im Gegensatz zu den anderen nur bedienbar, wenn man den Territorium-Editor selektiert hat. Verliert der Territorium Editor den Fokus, so ist das Actionset zwar sichtbar aber dessen Funktionen werden deaktiviert (grau dargestellt). Wird der Territorium-Editor von einem anderen Editor überdeckt oder wird er geschlossen, so verschwindet auch das Hamster-Territorium-Actionset.

Hat man den Hamster-Editor geöffnet und diesen selektiert, so wird der Standard-Territorium-Button (Mauer Symbol) in der globalen Toolbar verfügbar, welcher das Standard-Territorium im Territorium-Editor anzeigt.

Die restlichen Buttons werden hier nicht behandelt, da sie nicht für das Arbeiten mit dem Hamster-Simulator-Plugin notwendig sind. Des Weiteren können einzelne Actionsets auch deaktiviert werden, andere zusätzlich aktiviert werden und in ihrer Reihenfolge beliebig angeordnet werden, so das nicht zwangsläufig die oben beschriebene Reihenfolge von Actionsets eingehalten werden muss.

### 4.1.2 Die Hamster-Wizards

Die Hamster-Wizards sind hilfreiche Programme die den Benutzer dabei unterstützen neue Hamster-Projekte, Hamster-Dateien und Hamster-Territorien anzulegen. Um einen der drei Wizards zu öffnen, muss das Menü „New“ geöffnet werden, welches von den verschiedensten Stellen aus erreichbar ist. Dies geht z.B. über das Menü „File“ den „New“-Button und dessen „Drop-Down“-Menü oder über das Kontextmenü des Navigator-Views. Im Menü „New“ können Sie den gewünschten Hamster-Wizard wählen. Die einzelnen Wizards werden näher in den Abschnitten 4.2.1, 4.3.1 und 4.4.1 beschrieben.

### 4.1.3 Der Navigator View

Der „Navigator View“ (sh. Abbildung 4.2) stellt die zentrale Dateiverwaltung im Hamster-Simulator-Plugin dar und ist als Gegenstück zum Dateibaum des „Hamster Simulators“ in der Version 2.2 zu sehen. Der „Navigator View“ zeigt den Inhalt des Workspace-Verzeichnisses, welches beim ersten Start festgelegt wurde. In diesem Verzeichnis werden alle Dateien die in Eclipse erstellt werden gespeichert, wobei diese in Projekten gruppiert werden, welche wiederum Verzeichnisse im Workspace-Verzeichnis darstellen. In Projekten können Dateien und Ordner in einer „gewissen“ Schachtelungstiefe angelegt werden, wobei diese im „Navigator View“ als Baumstruktur dargestellt werden. Hamsterdateien werden im Hamster-Simulator-Plugin im Workspace in so genannten Hamster-Projekten angelegt. Diese werden im „Navigator View“ als Ordner-Symbol mit der Aufschrift „HAM“ visualisiert. Es gibt fünf verschiedene

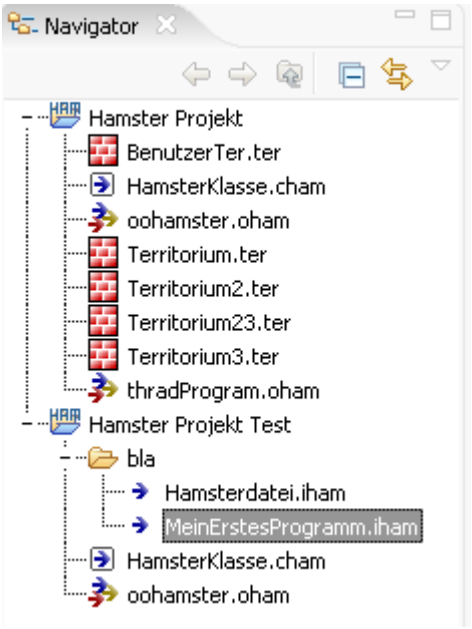


Abbildung 4.2: Navigator View

Arten von Hamsterdateien im Hamster-Simulator-Plugin, welche in Hamster-Projekten angezeigt werden. Dies sind die imperativen und objektorientierten Hamster-Programme, die Hamster-Klassen, die Hamsterdateien aus dem bisherigen „Hamster Simulator“ mit der Dateiendung „*.ham“ sowie die Territoriendateien. Zusätzlich befinden sich in Hamster-Projekten noch zu jedem Hamster-Programm und jeder Hamster-Klasse Dateien mit der Endung *.java und *.class. Diese werden aber gefiltert, so dass Sie nicht angezeigt werden. Der „Navigator View“ verfügt über eine lokale Toolbar, in welcher weitere Funktionen verfügbar sind. Diese werden ausführlich in der Eclipse Hilfe erläutert. In Abbildung A.1 ist der „Navigator View“ mit den verschiedenen Hamster-Dateitypen dargestellt.

Als zentrale Dateiverwaltung ist es im „Navigator View“ möglich Dateien und Ordner umzubenennen, zu verschieben, zu kopieren und zu löschen. Dies Kann durch „Shortcuts“, per „Drag and Drop“ oder über das Kontextmenü erfolgen. Ferner ist es möglich Dateien per Doppelklick oder über das Kontextmenü (open) zu öffnen. Im Kontextmenü besteht zusätzlich die Möglichkeit die selektierte Hamster- oder Territoriendateien mit dem Eintrag „open with“ mit einem anderen als dem jeweiligen Standard-Editor zu öffnen. Dabei ist darauf zu achten, dass diese Datei zukünftig immer mit dem gewählten Editor geöffnet wird, sofern nicht über den „open with“ Eintrag im Kontextmenü der Standard-Editor wieder ausgewählt wird.

Selektiert Hamster-Programme können über ihr Kontextmenü im „Navigator View “ im „run“ oder „debug“-Modus ausgeführt werden, wenn man „Run/Debug As→Hamsterprogramm“ wählt.

### 4.1.4 Der Territorium-Editor

Der Territorium-Editor (sh. Abbildung 4.3) stellt das Gegenstück zum Simulationsfenster im bisherigen „Hamster Simulator“ dar und hat zum einen die Aufgabe das Territorium darzu-

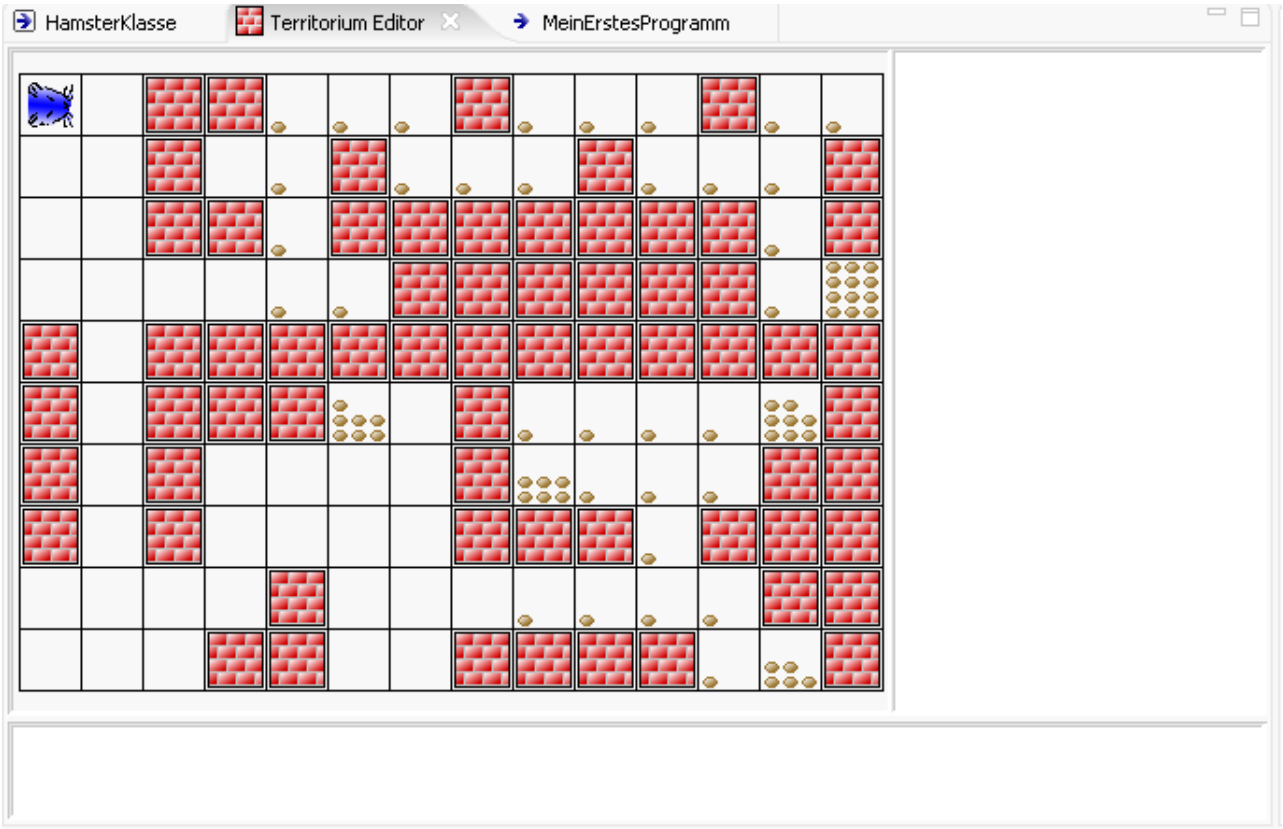


Abbildung 4.3: Territorium-Editor

stellen und dessen Modellierung zu unterstützen und es hat zum anderen die Aufgabe die Simulation des Hamster-Programms zu visualisieren. Anders als im bisherigen „Hamster Simulator“ wird der Territorium-Editor zusammen mit anderen Editoren in der Editor-Area angeordnet. Dies ist ein Bereich der speziell für Editoren vorgesehen ist und alle geöffneten Editoren beherbergt. Dabei können sich die Editoren gegenseitig überdecken, so das nur deren Tab („Reiter“) sichtbar ist, oder Sie können auch horizontal und vertikal zueinander angeordnet werden. Dies ist in der Hamster-Debug-Perspektive der Fall, wo der Hamster-Editor links von dem Territorium-Editor angeordnet ist. Um die Anordnung zu ändern kann man einfach den entsprechenden Editor am Tab markieren, die Maustaste gedrückt halten und an die gewünschte Position ziehen (funktioniert auch mit Views).

Ähnlich wie im bisherigen „Hamster Simulator“ kann es nur ein geöffnetes Territorium gleichzeitig geben, so dass beim Öffnen eines neuen Territoriums im Territorium-Editor das bisher geöffnete Territorium verworfen wird. Da es keine implizite Speicherung des Territoriums gibt, muss man das Territorium über den „Save As“-Dialog („File→Save As“) explizit speichern, um das Territorium später wieder verwenden zu können. Gespeicherte und neu erzeugte Territorien werden im „Navigator View“ im jeweiligen Hamster-Projekt dargestellt und können per Doppelklick (Windows) oder über das Kontextmenü der selektierten Territoriumdatei geöffnet werden (rechte Maustasten→open).

Neben explizit erzeugten und gespeicherten Territorien gibt es das Standard-Territorium, wel-

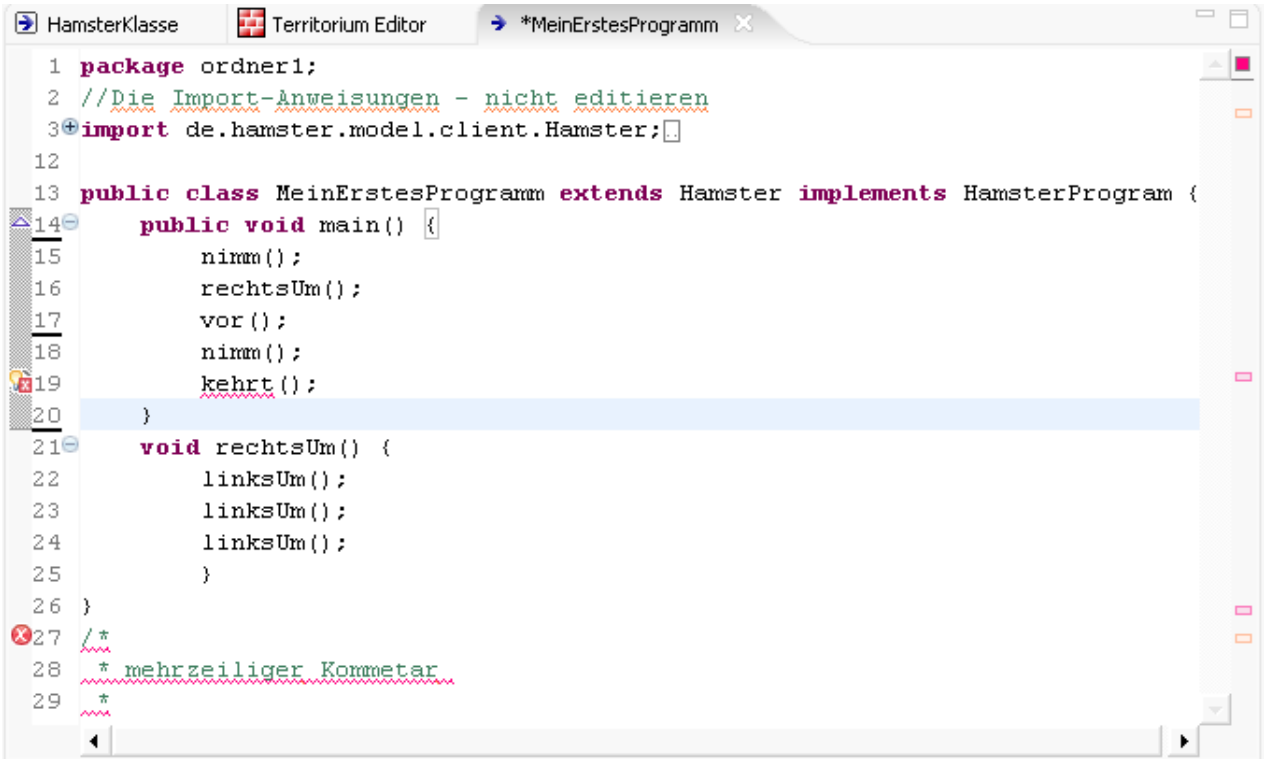


Abbildung 4.4: Hamster-Editor

ches immer das zuletzt bearbeitete Territorium anzeigt, bzw. beim Start von Eclipse ein leeres Territorium der Größe 14x10 zeigt. Das Standard-Territorium wird im Territorium-Editor gezeigt, wenn eine Hamster-Datei im Hamster-Editor geöffnet wird und kein Territorium-Editor geöffnet ist, oder wenn die Ausführung eines Hamster-Programms veranlasst wird und kein Territorium geöffnet ist. Man kann das Standard-Territorium auch explizit öffnen, indem man auf den Standard-Territorium-Button (Mauer-Symbol) klickt, welcher in der globalen Toolbar verfügbar ist, sobald der Hamster-Editor selektiert ist.

Bei selektiertem Territorium-Editor wird das Hamster-Territorium-Actionset in der globalen Toolbar verfügbar. Dieses stellt Funktionen bereit, um das Territorium zu editieren. Diese sind in Abschnitt 3.2 bereits aufgeführt.

### 4.1.5 Der Hamster-Editor

Der Hamster-Editor (sh. Abbildung 4.4) dient dem Schreiben von Hamster-Programmen im Hamster-Simulator-Plugin. Der Hamster-Editor ist als Standard-Editor für imperative und objektorientierte Hamsterdateien sowie Hamster-Klassen definiert, so dass dieser beim Erzeugen und Öffnen dieser Hamsterdateien geöffnet wird. Dabei können mehrere Instanzen des Hamster-Editors mit unterschiedlichen Hamsterdateien gleichzeitig geöffnet sein und sofern diese nicht nebeneinander/übereinander angeordnet sind, wird immer nur ein aktiver Editor angezeigt. Von den anderen geöffneten Editoren ist dann der Tab zu sehen, oder wenn der Platz nicht ausreicht kann auf das Doppelpfeil-Symbol rechts von den Editor-Tabs (sh. A.1)

geklickt werden, um sich die Liste der geöffneten Editoren anzeigen zu lassen.

Der Hamster-Editor ist zusätzlich in der Lage auch die Dateien des bisherigen „Hamster Simulators“ (Version 2.2) zu öffnen. Dabei werden diese Dateien vom Hamster-Editor in korrespondiere Hamsterdatein des Hamster-Simulator-Plugins umgewandelt und geladen. Die „*.ham“-Dateien werden gelöscht. Jedoch besteht die Möglichkeit auch Hamsterdateien des Hamster-Simulator-Plugins zurück in „*.ham“-Dateien zu konvertieren (vgl. 4.8.2).

Der Hamster-Editor öffnet anderes als der Editor des Editors des „Hamster Simulator“ (Version 2.2) nicht direkt die Hamsterdatei, sondern wandelt diese bei seinem Start in eine korrespondierende Javadatei um und zeigt diese an. Daher werden auch „import“-Anweisungen und Klassenrümpfe angezeigt. Für Sie als Benutzer macht es keinen Unterschied, ob Sie die Hamsterdatei oder die korrespondierenden Javadatei bearbeiten, da beim Speichern von Javadateien im Hamster-Editor die korrespondierenden Hamsterdateien geschrieben werden und wie gesagt beim öffnen der Hamsterdateien im Hamster-Editor die zugehörigen Javdateien erzeugt werden. Der Vorteil des Bearbeitens von den korrespondierenden Javadateien, anstelle der eigentlichen Hamsterdateien, liegt darin, dass der Hamster-Editor viele zusätzliche Funktionen zur Verfügung stellt. Dies sind neben den aus dem alten „Hamster Simulator“ bekannten Funktionen wie „Undo“, „Redo“, „Ausschneiden“, „Kopieren“, „Einfügen“ und „Syntax Highlighting“(Hervorhebung von Schlüsselwörten), vor allem die Funktionen: „automatische Code-Vervollständigung“, „Überprüfung des Codes auf syntaktische Fehler zur Laufzeit“, „Unterstützung von Markern“, „Unterstützung von Templates“, „Code-Formatierung“ und „explizites Erzeugen ausgewählter Funktionen“ sowie „Finden von Referenzen und Deklarationen“, „Anzeigen des Javadocs von Methoden als Tooltip“, „Ausblenden von bestimmten Programmteilen“ und „die Anzeige einer Gliederung im Outline View“. Alle diese zusätzlichen Funktionen werden im Abschnitt 4.3.10 näher erläutert.

### 4.1.6 Der Outline View

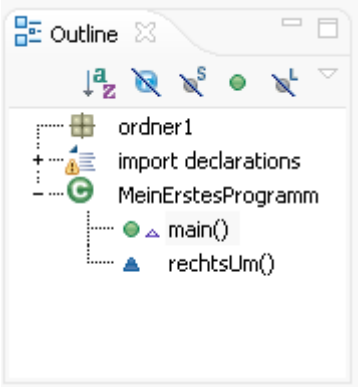


Abbildung 4.5: Outline View

Der „Outline View“ (sh. Abbildung 4.5) ist eine Komponente des „Eclipse SDK“ und ist in der Lage die Gliederung eines Dokuments eines markierten Editors anzuzeigen. Für Hamsterdateien die im Hamster-Editor geöffnet sind bedeutet dies, dass Sie im „Outline View“ nach Imports, Variablen, Konstruktoren und Methoden gegliedert dargestellt werden. Dabei zeigen Symbole an, ob die Methoden und Attributen vom Typ „public“, „protected“ oder „private“

sind und ob diese statisch oder nicht statisch sind. Der „Outline View“ bietet noch eine reihe Weiterer Funktionen, wie beispielsweise die Anzeige von ausführbaren Klassen oder die Anzeige von internen Klassen, uvm. Für weitere Informationen sei daher auf die Eclipse Hilfe verwiesen.

Selektiert man ein Element im „Outline View“, so springt Eclipse automatisch zu dem jeweiligen Element im entsprechenden Editor.

### 4.1.7 Der Problems View

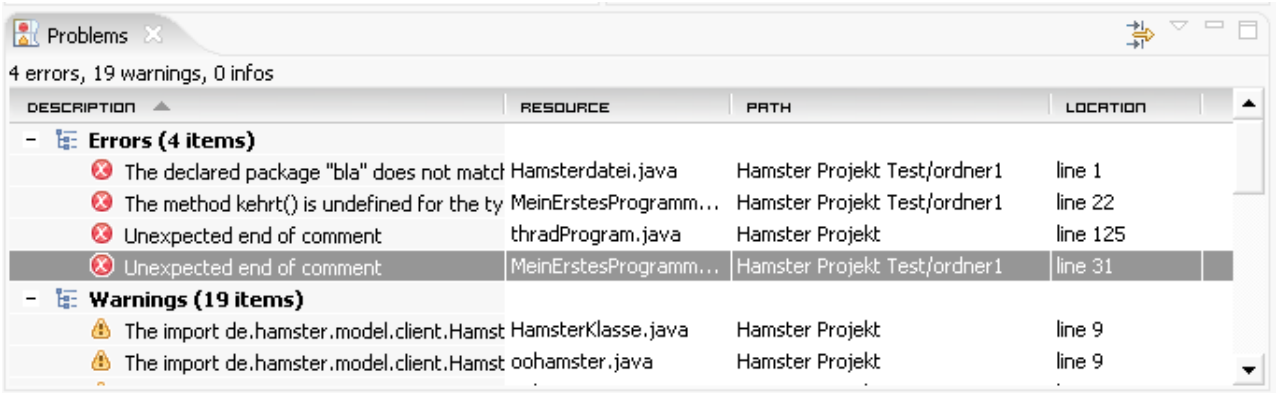


Abbildung 4.6: Problems View

Der „Problems View2 (sh. Abbildung 4.6) ist eine Eclipse-Komponente und wird vom Hamster-Simulator-Plugin verwendet. Er zeigt grundsätzlich zwei Arten von Fehlermeldungen. Zum einen sind dies „Error“-Meldungen, also syntaktische Fehler, die im Quellcode vorhanden sind und zum anderen „Warning“-Meldungen. „Warning“-Meldungen sind nicht schwerwiegend und Programme können normal ablaufen, jedoch weisen Sie darauf hin, dass beispielsweise deklarierte „import“-Anweisungen oder Variablen nicht verwendet werden.

Die Fehler sind in tabellarischer Form im „Problems View“ angegeben, wobei die erste Spalte das Symbol des Fehlertyps zeigt. Die zweite Spalte zeigt eine Beschreibung des Fehlers und gibt meistens Aufschluss darüber wie der Fehler behoben werden kann. Die dritte und vierte Spalte zeigt die Datei und den Ordner, in der der Fehler aufgetreten ist. In der fünften Spalte ist schließlich die genaue Zeile angegeben, die den Fehler beinhaltet.

Durch Doppelklick mit der linken Maustaste kann zu dem Fehler gesprungen werden. Alternativ kann man dies auch durch den Eintrag „Go To“ im Kontextmenü des selektierten Fehlers erreichen. Der Eintrag „Quick Fix“ im Kontextmenü des selektierten Fehlers erlaubt es, den Fehler von Eclipse beseitigen zu lassen. Allerdings ist dies nicht bei allen Fehlern möglich.

Grundsätzlich werden alle Fehler von Dateien im Workspace angezeigt. Durch Filter ist es jedoch auch möglich, die angezeigten Fehler auf beispielsweise das lokale Projekt einzuschränken. Die Filteroptionen können wie in Abb. A.1 zu sehen über das Filter-Symbol der lokalen Toolbar des „Problem Views“ erreicht werden.

Weitere Informationen zum „Problems View“ können in der Eclipse Hilfe nachgelesen werden.

### 4.1.8 Debug View

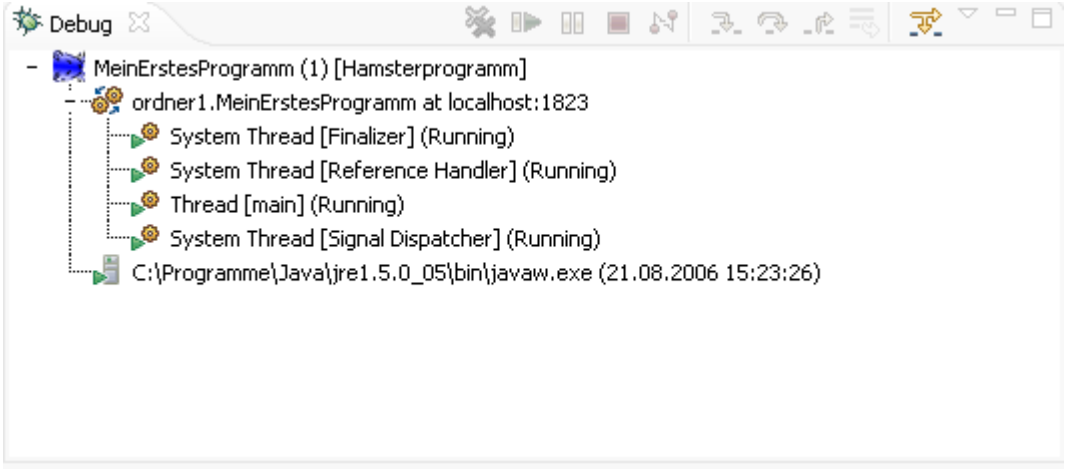


Abbildung 4.7: Debug View

Der „Debug View“ (sh. Abbildung 4.7) ist eine Eclipse-Komponente und ermöglicht unter anderem das debuggen von Hamster-Programmen, d.h. die schrittweise Ausführung des Programmcodes. Der „Debug View“ wird standardmäßig in der Hamster-Debug-Perspektive in der linken oberen Ecke angezeigt. Er wird aber auch angezeigt, wenn man in der Hamster-Perspektive ein Hamster-Programm im „debug“-Modus startet und dieses pausiert.

Der „Debug View“ zeigt in einer Baumstruktur als Wurzelelement den Namen des Programms und den Typ des Programms (Hamster-Programm). In der nächsten Ebene des Baums zeigt das erste Element wiederum den Namen des Programms, den Server, auf dem das Programm läuft (localhost= lokaler Rechner) und den Port. Man spricht bei diesem Element von dem so genannten „Debug Target“. Als zweites Element sieht man den Namen der „Virtual Machine“, die das Programm ausführt sowie das Datum und die Uhrzeit, zu der die Ausführung begonnen hat. Das „Debug Target“ verfügt über weitere Kind-Elemente. Dies sind die jeweiligen „Threads“ die von dem Programm genutzt werden und in der „Virtual Machine“ ablaufen. Selektiert man das „Debug Target“ und wählt den Pause-Button, so werden alle „Thread“ pausiert (Sandard-Einstellung im Hamster-Simulator-Plugin wenn nichts selektiert). Selektiert man einen einzelnen „Thread“ und wählt den Pause-Button, so wird nur dieser pausiert. Pausierte „Threads“ haben als Kindelement den so genannten „Callstack“. Dieser zeigt die Methoden in der Reihenfolge, wie Sie vom Thread im Programmablauf aufgerufen werden (von unten nach oben), wobei auch die Klasse, in der die Methode sich befindet und die Zeilenzahl angegeben werden. Selektiert man eine Methode, so zeigt Eclipse die Klasse mit der jeweiligen Methode in einem Editor an, sofern Source-Code zu der Klasse verfügbar ist.

Der „Debug View“ erlaubt die zeilenweise Ausführung von Programm-Code, wenn das Programm pausiert ist. Dazu muss man die Methode selektieren, dessen weitere Fortführung man genauer betrachten will. Dabei stehen einem die drei Pfeiltasten „Schritt hinein“ (Step Into), „Schritt hinüber“ (Step Over) und „Schritt hinaus“ (Step Return) zur Verfügung, wobei mit „Schritt hinein“ in den Methodenrumpf der auszuführenden Methode springen kann, mit „Schritt hinüber“ die auszuführende Methode (und alle weiteren implizit aufgerufenen Methoden) ausführen kann ohne in den Methodenrumpf zu springen und mit „Schritt hinaus“ den

aktuellen Methodenrumpf verlassen und zur aufrufenden Methode zurückkehren kann.

Mit Hilfe des Fortsetzen-Buttons kann die Programm-Ausführung fortgesetzt werden und mit Hilfe des Stopp-Buttons kann die Ausführung des Programms abgebrochen werden.

Für weitere Informatioen zum „Debug View“ sei auf die Eclipse Hilfe verwiesen.

### 4.1.9 Variables View

Abbildung 4.8: Variables View

Der „Variables View“ (sh. Abbildung 4.8) ist eine Eclipse-Komponente, die in der Lage ist die Variablenbelegungen einer Klasse während dessen Ausführung zu zeigen. Dafür muss die Klasse von einer „Virtual Machine“ in einem Thread ausgeführt werden und der Thread muss pausiert werden. Nun geht man über den Callstack in die Klasse, dessen Variablenbelegung man sehen möchte. Der „Variables View“ unterscheidet zwischen lokalen und globalen Variablen und dem Datentyp der Variablen. Globale Variablen werden unter der Variablen „this“ angezeigt. Lokale Variablen werden auf der gleichen Stufe wie „this“ angezeigt. Ist die Variable von einem einfachen Datentyp (int, long, char, etc.) so wird nur dessen Belegung angezeigt. Handelt es sich hingegen um eine Variable mit einem komplexen Datentyp (String, Hamster, etc.), so wird in einer Baumstruktur deren interne Variablen, die auch wieder aus einfachen und komplexen Datentypen bestehen können, angezeigt.

Für weitere Informationen zum „Variables View“ sei auf die Eclipse Hilfe verwiesen.

## 4.2 Verwalten von Hamster-Projekten

In diesem Abschnitt wird nun detailliert vorgestellt, wie man Hamster-Projekte erzeugt, wie man sie verwaltet und welche Probleme im Zusammenhang mit Hamster-Projekten auftreten können.

### 4.2.1 Erzeugen von Hamster-Projekten

Um ein neues Hamster-Projekt im Workspace zu erzeugen müssen Sie den „Hamster Projekt Wizard“ starten. Dazu haben Sie in Eclipse verschiedene Möglichkeiten, wobei man zwischen dem Wizard-Auswahl-Dialog und Shortcuts unterscheidet:

Den Wizard-Auswahl-Dialog erreichen Sie wie folgt:

- Menü: File→New→Other
- „New“-Button in der globalen Toolbar
- Kontextmenü des „New“-Buttons→Other
- Kontextmenü des „Navigator Views“→Other

Dort können Sie den „Hamster Projekt Wizard“ in der Kategorie Hamster-Simulator wählen. Ebenso können Sie dort auch die anderen Hamster-Wizards wählen.

Die Shortcuts zum schnellen Laden des „Hamster Projekt Wizard“ erreichen Sie wie folgt (gilt nur für Hamster- und Hamster-Debug-Perspektive):

- Menü: File→New
- Kontextmenü „New“-Button
- Kontexxtmenü des „Navigator View“→New

Auf die gleiche Art und Weise können Sie auch die Shortcuts der anderen Hamster-Wizards erreichen.

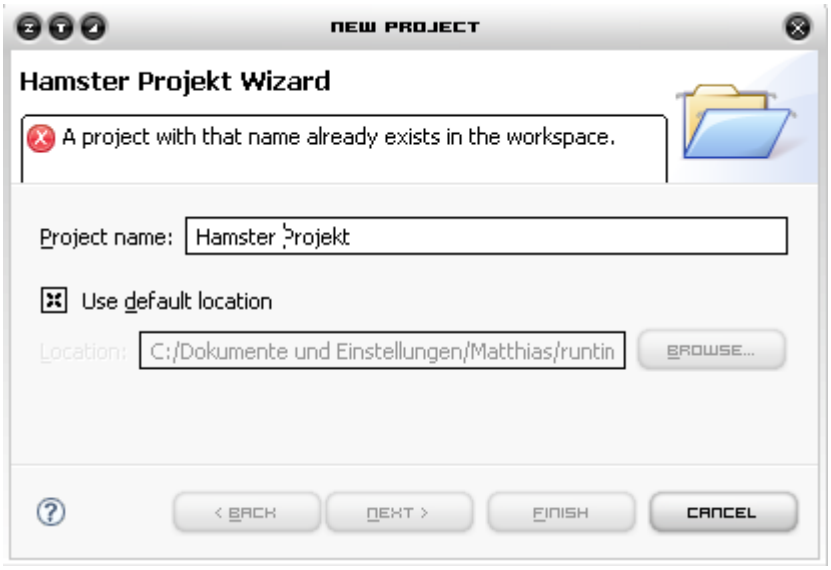


Abbildung 4.9: Hamster-Projekt-Wizard

Ist der „Hamster Projekt Wizard“ geöffnet, so müssen Sie wie in Abbildung 4.9 zu sehen einen neuen Namen für das Hamster-Projekt wählen. Dieser kann aus einem oder mehreren Wörtern bestehen, darf jedoch nicht bereits im Workspace verwendet werden. Wird der Name bereits verwendet, so erscheint im oberen Bereich des Wizards eine Fehlermeldung.

Alle Hamster-Projekte müssen im Workspace angelegt werden. Zwar gibt es die Möglichkeit

im Hamster-Projekt-Editor auch andere Ordner, die nicht im Workspace liegen, als Hamster-Projekte einzubinden, indem man eine andere als die „default Location“ wählt, jedoch wird davon dringend abgeraten, da das Starten von Hamster-Programmen, die nicht im Workspace liegen, zur Zeit vom Hamster-Simulator-Plugin nicht unterstützt wird.

Sie können den Wizard nun durch betätigen des „Finish“-Buttons beenden (empfohlen), durch betätigen des „Cancel“-Button abbrechen oder durch betätigen des „Next“-Buttons zur nächsten Seite gelangen, auf der weitere Einstellungen für das Hamster-Projekt vorgenommen werden können. Die nächste Seite ist äquivalent zu der zweiten Seite des „Java Projekt Wizards“, weshalb hier auf dessen Hilfe verwiesen wird.

### 4.2.2 Schließen und Öffnen von Hamster-Projekten

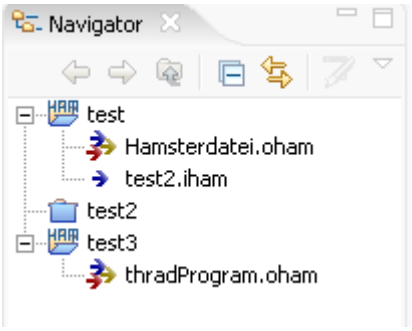


Abbildung 4.10: Geschlossenes Projekt im „Navigator View“

Wird ein Hamster-Projekt nicht verwendet, so besteht die Möglichkeit ein Hamster-Projekt zu schließen. Dies geschieht, indem Sie das Hamster-Projekt im „Navigator View“ selektieren und in dessen Kontext-Menü den Eintrag „Close Project“ wählen. Geschlossene Projekte werden wie das Projekt „test2“ in Abbildung 4.10 als geschlossener Ordner dargestellt und haben den Vorteil, dass sie beim „Build“ des Workspaces (Kompilieren aller Dateien im Workspace) nicht berücksichtigt werden und das Fehler in diesem Projekt nicht im „Problems View“ aufgelistet werden.

Das Öffnen von geschlossenen Hamster-Projekten erfolgt analog zum schließen von geöffneten Hamster-Projekten, indem Sie im Kontext-Menü des selektierten Hamster-Projekts den Eintrag „Open Project“ wählen.

### 4.2.3 Unterordner in Hamster-Projekten

Mit Hilfe des New-Folder-Wizards (zu erreichen über Menü File→ New→ Other→ General→ Folder) können auch Unterordner in Hamster-Projekten angelegt werden, welche wiederum Unterordner und Hamsterdateien aufnehmen können. Legt man Hamsterdateien in spezifischen Unterordnern ab, so spricht man im Kontext der Java-Programmierung auch von „Packages“, wobei diese Unterordner nur aus genau einem Wort bestehen dürfen! Werden Hamster-Programme oder Hamster-Klassen in diesem Unterordner erzeugt, so wird diesen automatisch der Eintrag `package Unterordner1/Unterordner2/...;` bzw. `package Unterordner1;` hinzugefügt.

### 4.2.4 Probleme mit Hamster-Projekten

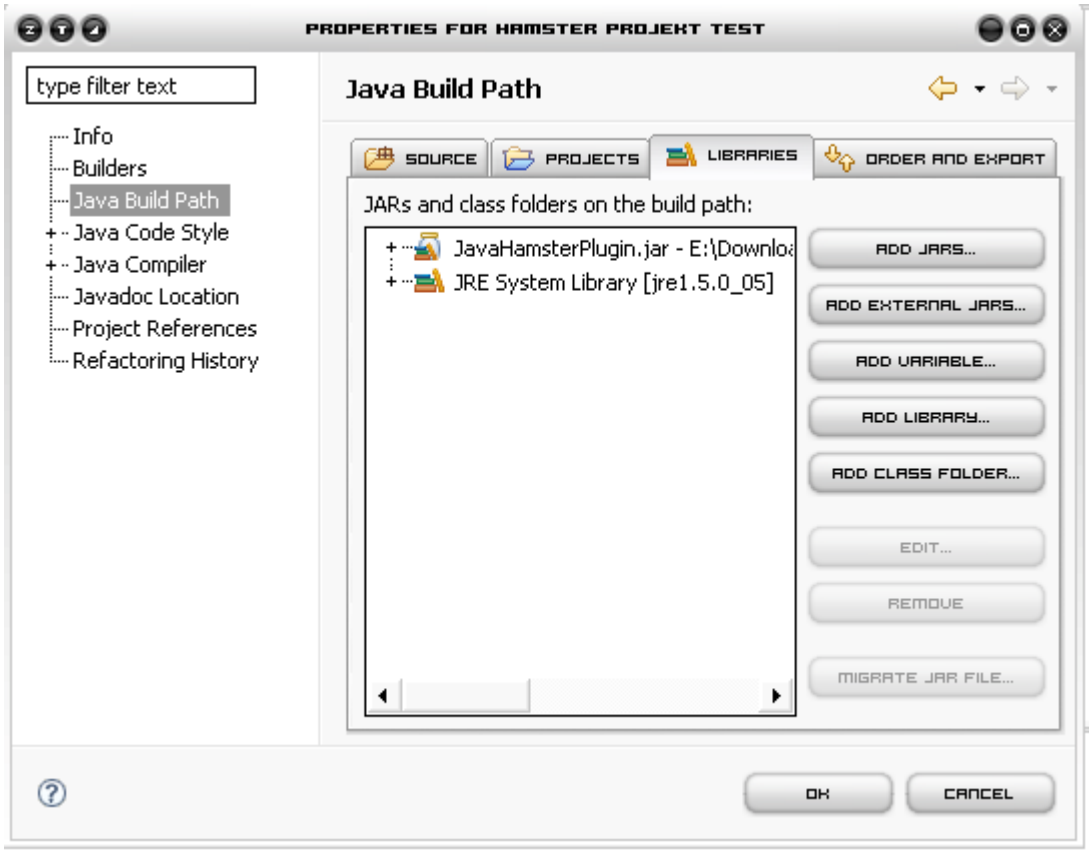


Abbildung 4.11: Projekt Eigenschaften

Wird ein Hamster-Projekt durch den „Hamster Projekt Wizard“ erzeugt, so werden auch dessen Eigenschaften gesetzt. Eine dieser Eigenschaften ist der „Build Path“, der unter anderem die vom Hamster-Simulator-Plugin benutzten Dateien angibt. Konkret ist durch diese Angabe das Importieren und Verwenden der Klassen `Hamster`, `Territorium` sowie der `Exceptions` in Hamsterdateien möglich. Es kann jedoch vorkommen, dass in bestimmten Systemen der „Build Path“ nicht richtig gesetzt wird. Dies ist kann insbesondere der Fall sein, wenn das Plugin in einem anderen Verzeichnis als dem Plugin-Verzeichnis der Eclipse-Installation installiert worden ist. Auch kann dieser Fehler möglicherweise mit anderen Betriebssystemen als „Windows XP“ auftreten.

Um den „Build Path“ für ein Hamster-Programm richtig zu setzen und somit sicher zu stellen, dass die Importanweisungen in den Hamsterdateien funktionieren, wählen Sie im Kontextmenü des Hamster-Projekts den Eintrag „Properties“ aus. Wählen Sie dann wie in Abbildung 4.11 zu sehen den Eintrag „Java Build Path“ auf der linken Seite aus und wechseln Sie anschließend auf der rechten Seite auf die Karteikarte „Libraries“. Entfernen Sie den Eintrag „JavaHamsterPlugin.jar“ da dieser Eintrag fehlerhaft ist. Dazu selektieren Sie den Eintrag und wählen den Button „Remove“ auf der rechten Seite des Fensters. Nun müssen Sie den korrekten Eintrag erzeugen. Dazu wählen Sie den Button „Add External Jars“ und wechseln in den Ordner, indem Sie das Hamster-Simulator-Plugin installiert haben. Gehen Sie in den

Plugin-Ordner hinein und wählen Sie die Datei „JavaHamsterPlugin.jar“ aus. Wählen Sie nun den Button „Öffnen“. Die Datei erscheint wie in Abbildung 4.11 zu sehen in der Liste mit den verfügbaren Bibliotheken. Betätigen Sie den „OK“-Button um den Vorgang zu beenden.

## 4.3 Verwalten und Editieren von Hamster-Dateien

### 4.3.1 Schreiben eines neuen Hamster-Programms

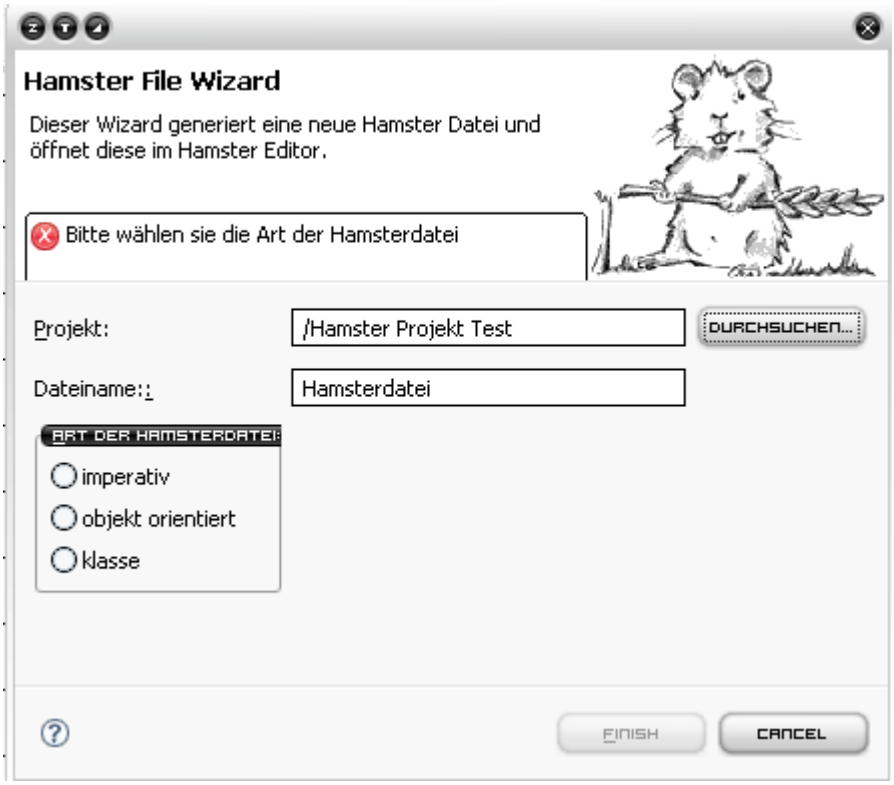


Abbildung 4.12: „Hamster File Wizard“

Um ein neues Hamster-Programm oder eine neue Hamster-Klasse zu schreiben müssen Sie den „Hamster File Wizard“ starten. Gehen Sie dazu analog zum Öffnen des „Hamster Projekt Wizards“ vor (vgl. 4.2.1).

Sie sehen nun den in Abbildung 4.12 dargestellten „Hamster File Wizard“. Zunächst müssen Sie nun ein Hamster-Projekt wählen, in dem Sie die neue Hamsterdatei erstellen wollen. Dann wählen Sie bitte einen Namen für die Hamsterdatei, wobei der gleiche Name nur einmal im Projekt verwendet werden darf. Eine Angabe der Dateiendung ist nicht notwendig! Als drittes müssen Sie noch die Art der Hamsterdatei wählen. Dazu wählen Sie den entsprechenden Radio-Button aus. Sofern nun keine Fehler angezeigt werden, wird der „Finish“-Button freigeschaltet, welchen Sie wählen müssen, um den Vorgang zu beenden. Werden noch Fehler im oberen Teil des Wizards angezeigt, so befolgen Sie bitte die dort angegebenen Schritte um den Fehler zu beheben.

Nach dem Betätigen des „Finish“-Buttons wird die Hamster-Datei in dem entsprechenden Hamster-Projekt neu angelegt und im Hamster-Editor zum Bearbeiten geöffnet. Sie können nun ihr Hamster-Programm schreiben, abspeichern (kompilieren) und ausführen.

### 4.3.2 Ändern eines existierenden Hamster-Programms

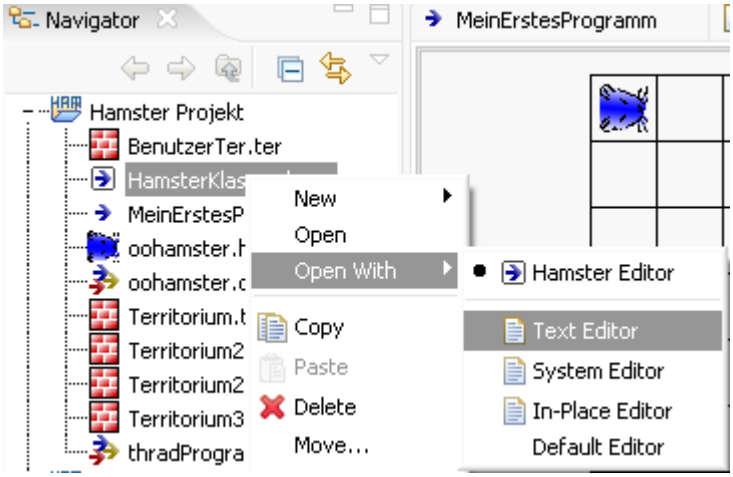


Abbildung 4.13: Hamsterdatei im anderen Editor öffnen

Um ein existierendes Hamster-Programm oder eine existierende Hamster-Klasse zu Ändern muss Diese geöffnet werden. Dazu kann im „Navigator View“ ein Doppelklick mit der linken Maustaste auf die entsprechende Hamsterdatei ausgeführt werden. Alternativ kann im Kontextmenü (rechte Maustaste) der selektierten Hamsterdatei der Eintrag „open“ gewählt werden um die Hamsterdatei in ihrem Standard-Editor zu öffnen, welches der Hamster-Editor ist. Es besteht auch die Möglichkeit die Hamsterdatei mit einem anderen Editor als dem Standard-Editor zu Bearbeiten. Dazu wählen Sie im Kontextmenü der selektierten Hamsterdatei, wie in Abbildung 4.13 zu sehen, den Eintrag „open with“ und wählen den entsprechenden Editor aus. Je nach gewähltem Editor stehen dann verschiedene Funktionen zum Bearbeiten der Hamsterdatei zur Verfügung. Wird eine Hamsterdatei mit einem anderen als den Standard-Editor geöffnet, so wird dieser Editor als Standard-Editor für diese Datei gewählt. Um wieder den Hamster-Editor als Standard-Editor der Datei zuzuweisen muss dieser im „Open with“-Dialog gewählt werden. Eine Hamsterdatei kann in einem speziellen Editor nur einmal geöffnet werden. Die Datei kann aber zur gleichen Zeit mit mehreren verschiedenen Editoren geöffnet werden.

Grundsätzlich sollten Hamsterdateien im Hamster-Editor bearbeitet werden da dieser die in Abschnitt 4.3.10 vorgestellten Funktionen zur Verfügung stellt. Des Weiteren ist nur dieser Editor in der Lage die zu einer Hamsterdatei gehörende Java- und Class-Datei zu erzeugen, ohne welche das Hamster-Programm nicht ablauffähig ist.

Ist die Hamsterdatei im Hamster-Editor geöffnet, so kann sie nach Belieben bearbeitet werden.

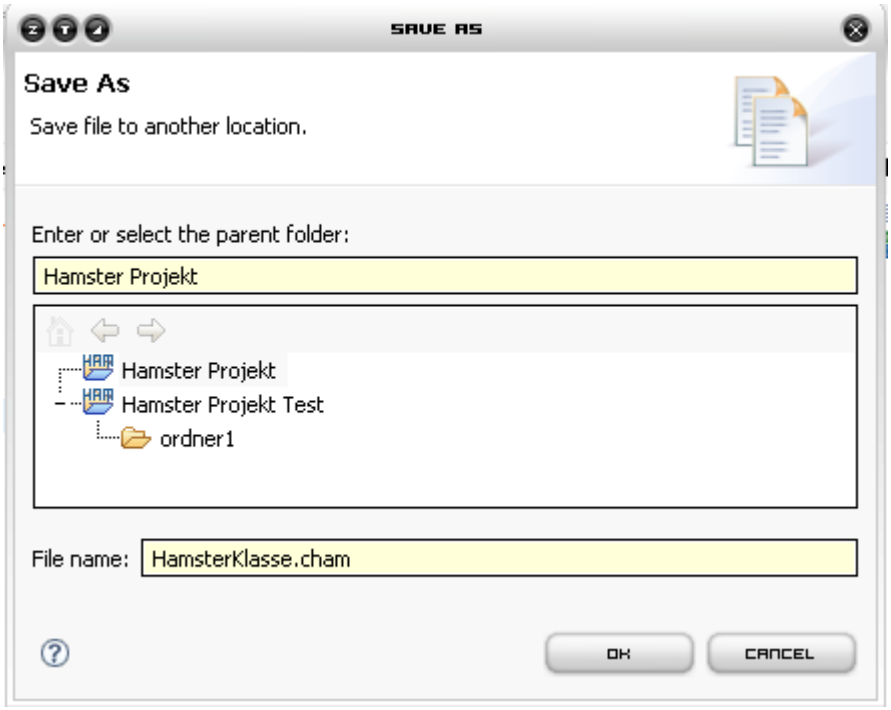


Abbildung 4.14: „Save As“-Dialog

### 4.3.3 Speichern eines Hamster-Programms

Ist das Speichern notwendig, so erscheint auf der linken Seite im Tab des Editors ein Stern-Symbol. Um ein Hamster-Programm oder eine Hamster-Klasse zu Speichern selektieren Sie den Editor und wählen im Menü „File“ den Eintrag „Save“. Alternativ können Sie auch die Tastenkombination „Ctrl+S“ wählen oder in der globalen Toolbar den „Speichern“-Button (Diskette-Symbol) betätigen.

Um den Editor-Inhalt in einer anderen Datei zu Speichen müssen Sie den „Save As“-Dialog öffnen. Dies machen Sie, indem Sie den zu speichernden Editor selektieren und dann im Menü „File“ den Eintrag „Save As“ wählen oder die Tastenkombination „Ctrl+Shift+S“ betätigen. In dem in Abbildung 4.14 abgebildeten „Save As“-Dialog müssen Sie das Projekt und den Dateinamen, unter dem die Datei gespeichert werden soll, angeben. Dabei ist es wichtig, dass Sie die Dateiendung mit angeben und diese zum Editor-Inhalt passt!

Wird ein Hamster-Programm oder eine Hamster-Datei vom Hamster-Editor aus gespeichert, so werden nicht nur die Hamster-Dateien sondern auch die Java-Dateien gespeichert. Zusätzlich findet beim Speichern das Kompilieren der Hamsterdateien statt und es werden die zugehörigen Class-Dateien erzeugt, die zum Ablaufen von Hamster-Programmen notwendig sind. Treten beim Kompilieren Probleme auf, da das Hamster-Programm oder die Hamster-Klasse syntaktische Fehler enthält, so werden diese im „Problems View“ dargestellt.

### 4.3.4 Löschen eines existierenden Hamster-Programms

Um ein existierendes Hamster-Programm oder eine Hamster-Klasse zu löschen muss die Hamsterdatei im „Navigator View“ selektiert werden. Durch Betätigen der „Entf“-Taste wird der „Confirm Resource Delete“-Dialog aufgerufen, mit welchem man die Datei endgültig löschen kann oder den Löschen-Vorgang abbricht. Dieser Dialog kann alternativ auch über das Kontextmenü der selektierten Hamsterdatei und den Eintrag „Delete“ aufgerufen werden. Die dritte Möglichkeit den „Confirm Resource Delete“-Dialog zu öffnen, besteht darin, die Hamsterdatei zu selektieren und Menü Edit→Delete zu wählen.

Beim Löschen werden sowohl die Hamsterdatei, als auch die zugehörigen Java- und Class-Dateien gelöscht und der Hamster-Editor mit der Datei geschlossen, falls er noch geöffnet sein sollte.

### 4.3.5 Umbenennen eines existierenden Hamster-Programms

Um ein Hamster-Programm oder eine Hamster-Klasse umzubenennen gibt es verschiedene Möglichkeiten. Diese sind selektieren der Hamsterdatei, drücken von „F2“ oder wählen von Menü „File→Rename“ oder wählen des Eintrags „Rename“ im Kontextmenü. Nun kann die Datei umbenannt werden, wobei die Dateiendung nicht umbenannt werden sollte.

War der Hamster-Editor mit dieser Datei geöffnet, muss der Klassenrumpf umgeändert werden. Alternativ kann man den entsprechenden Hamster-Editor vor dem Umbenennen auch schließen und nach dem Umbenennen wieder öffnen.

### 4.3.6 Verschieben eines existierenden Hamster-Programms

Um ein Hamster-Programm oder eine Hamster-Klasse zu verschieben gibt es verschiedene Möglichkeiten: Der einfachste Weg ist per „Drag and Drop“, was bedeutet, dass man die Datei selektiert, die (linke) Maustaste gedrückt hält, zu dem Projekt/Ordner mit der Maus geht, in die die Hamsterdatei verschoben werden soll und dort die (linke) Maustaste loslässt. Eine andere Möglichkeit ist das Selektieren der Hamsterdatei und wählen des Eintrags „Move“ im Kontextmenü der Datei oder im Menü „File“. In dem nun erscheinenden Dialog „Folder Selection“ wählt man das Hamster-Projekt aus, in welches die Hamsterdatei verschoben werden soll.

Man kann auch die Dateien mit den oben genannten Mechanismen in andere Verzeichnisse als das Workspace verschieben, indem man beispielsweise den „Explorer in Windows“ zu Hilfe nimmt.

Ist die zu verschiebende Datei in einem Hamster-Editor geöffnet, so wird dem Hamster-Editor automatisch die neue Position der Hamsterdatei mitgeteilt.

Wird die Hamsterdatei in einen Unterorder eines Projektes verschoben, so muss eine `package`-Anweisung in das Hamster-Programm / Hamster-Klasse geschrieben werden. Diese befindet sich vor den Import-Anweisungen und hat die Form `package Unterordner1/Unterordner2/...;` bzw. `package Unterordner1;`

### 4.3.7 Kopieren eines existierenden Hamster-Programms

Um ein Hamster-Programm oder eine Hamster-Klasse zu kopieren kann wie beim Verschieben „Drag and Drop“ verwenden. Dazu selektiert man im „Navigator View“ die entsprechende Hamsterdatei und hält die (linke) Maustaste gedrückt. Gleichzeitig hält man auch die „Ctrl“-Taste gedrückt (abhängig vom System) und geht mit der Maus zu dem Hamster-Projekt/Ordner in welchen man die Hamsterdatei kopieren will. Darüber lässt man die (linke) Maustaste los und die Hamsterdatei wird kopiert. Um die Hamsterdatei zu kopieren kann man aber auch die Hamsterdatei selektieren und im Kontextmenü oder Menü „Edit“ den Eintrag „Copy“ wählen. Alternativ ist auch die Tastenkombination „Ctrl+C“ möglich. Um die Datei dann in einem anderen Hamster-Projekt/ Ordner einzufügen wählt man im Kontextmenü oder Menü „Edit“ den Eintrag „Paste“ bzw. wählt alternativ die Tastenkombination „Ctrl+V“.

Es ist auch möglich die Dateien aus dem Workspace über die oben genannten Mechanismen in Verzeichnisse zu kopieren die nicht im Workspace liegen. Beispielsweise kann man per „Drag and Drop“ Dateien aus dem Workspace in Verzeichnisse im „Explorer“(Windows) ziehen und sie somit dorthin kopieren. Dabei entspricht das „Drag and Drop“ Verhalten dem im jeweiligen System zugrunde liegenden Regeln. Anders herum ist es natürlich auch möglich, Dateien in den Workspace zu kopieren. Gegebenenfalls muss dann der „Navigator View“ aber mit „F5“ aktualisiert werden.

Kopiert man Hamsterdateien in Unterordner so muss, wie beim Verschieben, die `package`-Anweisung angepasst oder neu hinzugefügt werden.

### 4.3.8 Drucken eines Hamster-Programms

Um ein Hamster-Programm oder eine Hamster-Klasse zu Drucken muss die zu druckende Hamsterdatei geöffnet und der jeweilige Editor selektiert sein. Nun wählen Sie den in Abbildung 4.1 dargestellten zweiten Button von links, der ein Drucker-Symbol darstellt. Dieser öffnet das jeweils systemspezifische Drucker-Menü mit dessen Hilfe Sie die Hamsterdatei drucken können. Alternativ können Sie auch im Menü „File“ den Eintrag „Print“ oder die Tastenkombination „Ctrl+p“ wählen um das systemspezifische Drucker-Menü zu öffnen.

### 4.3.9 Schließen eines geöffneten Hamster-Programms

Um ein Hamster-Programm oder eine Hamster-Klasse zu schließen, verweilen Sie mit der Maus über dem Tab des jeweiligen geöffneten Editors. Es erscheint auf der rechten Seite des Tabs ein Kreuz, auf welches Sie mit der (linken) Maustaste drücken, um den Editor zu schließen. Alternativ können Sie auch den Editor, den Sie schließen wollen selektieren in dessen Kontextmenü (sh. Abb. 4.15) oder im Menü „File“ den Eintrag „Close“ wählen. Daneben können Sie auch die Tastenkombination „Ctrl+W“ wählen um den selektierten Editor zu schließen.

Ist der Editorinhalt des zu schließenden Editors noch nicht gespeichert, so erscheint der „Save Resource“-Dialog, welcher Sie zum Speichen auffordert. Sie haben nun die Möglichkeit die Datei zu speichern (yes), nicht zu speichern (no) oder den Schließvorgang abzubrechen (cancel).

Um alle geöffneten Editoren in der Editor-Area zu schießen können Sie das Kontextmenü des

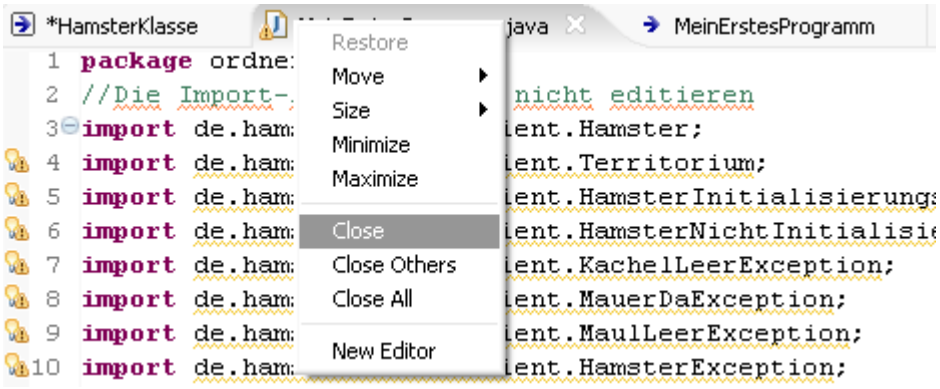


Abbildung 4.15: Schließen eines Editors

Tabs (rechte Maustaste) eines Editors aufrufen und den Eintrag „Close All“ wählen. Alternativ können Sie auch den Editor selektieren und im Menü „File“ den Eintrag „Close All“ bzw. die Tastenkombination „Ctrl+Shift+W“ wählen. Sind dabei Ressourcen noch nicht gespeichert, so erscheint der oben beschriebene „Save Resource“-Dialog.

Sie haben auch die Möglichkeit, alle bis auf den selektierten Editor zu schließen. Dazu müssen Sie im Kontextmenü des Tabs des Editors den Eintrag „Close Others wählen“.

### 4.3.10 Editor-Funktionen

Der Hamster-Editor verfügt über die verschiedensten Funktionen um Hamster-Programme und Hamster-Klassen komfortabel zu bearbeiten. Diese werden im folgenden näher erläutert:

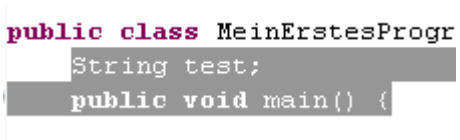


Abbildung 4.16: Markierter Code

- **Markieren von Code** - Um Code zu markieren (sh. Abbildung 4.16) gibt es verschiedene Möglichkeiten: Sie können zur Anfangsposition mit der Maus gehen, die Maustaste gedrückt halten und zur Endposition gehen. Sie können mit den Pfeiltasten der Tastatur zur Anfangsposition gehen und die Shift-Taste gedrückt halten und zur Endposition gehen. Alternativ können Sie, wenn Sie ein bestimmtes Wort markieren wollen auch ein Doppelklick mit der (linken) Maustaste auf dieses Wort ausführen.
  Um den gesamten Code zu markieren sollten Sie das Tastenkombination „Ctrl+A“ verwenden, oder im Menü “Edit“ den Eintrag „Select All“ wählen.
- **Code kopieren** - Wenn Sie Code kopieren wollen, müssen Sie diesen zunächst markieren und dann im Menü „Edit“ (sh. Abb. 4.17) oder im Kontextmenü des Editors den Eintrag „Copy“ wählen. Alternativ können Sie auch die Tastenkombination „Ctrl+C“ wählen. Der Code befindet sich nun in der Zwischenablage und kann an beliebiger Stelle eingefügt werden.

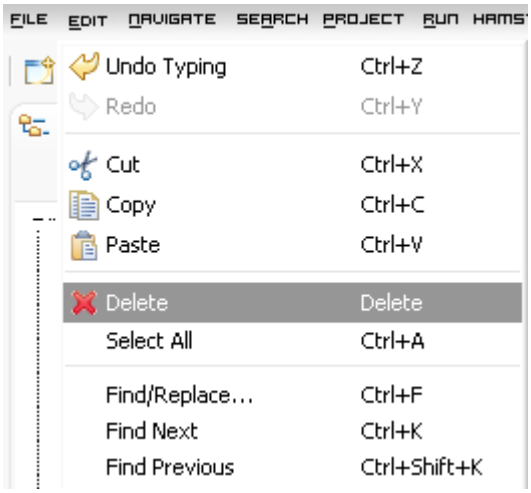


Abbildung 4.17: Teile des Menü „Edit“

- **Code ausschneiden** - Um den Code auszuschneiden, müssen Sie ihn zunächst selektieren und dann im Menü „Edit“ oder im Kontextmenü des Editors den Eintrag „Cut“ wählen. Alternativ können Sie auch ide Tastenkombination „Ctrl+X“ wählen.
- **Code löschen** - Den Code löschen Sie, indem Sie diesen markieren und die Taste „Entf“ bzw. „Del“ drücken oder im Menü „Edit“ den Eintrag „Delete“ wählen.
- **Code einfügen** - Code können Sie aus der Zwischenablage einfügen, indem Sie den Curser mit der Maus oder der Tastatur an die Position bringen, wo der Code eingefügt werden soll. Wählen Sie nun im Menü „Edit“ oder im Kontextmenü des Editors den Eintrag „Paste“. Alternativ können Sie auch die Tastenkombination „Ctrl+V“ wählen.
- Code-Aktion rückgängig machen - Um eine Code-Aktion rückgängig zu machen wählen Sie im Menü „Edit“ oder im Kontextmenü des Editors den Eintrag „Undo (Typing)“ oder die Tastenkombination „Ctrl+Z“.
  Es wird nur eine gewisse Anzahl an Aktionen gespeichert. Um die Anzahl zu verändern muss man die Eclipse Einstellungen ändern.
- **Code Aktion wieder ausführen** - Um eine Code-Aktion wieder auszuführen, nachdem Sie rückgängig gemacht wurde wählen Sie im Menü „Edit“ oder im Kontextmenü des Editors den Eintrag „Redo (Typing)“ oder die Tastenkombination „Ctrl+Y“.
- **Code Finden oder Ersetzen** - Selektieren Sie den Editor in dessen Inhalt Sie suchen wollen. Um den „Find or Replace“-Dialog zu öffnen wählen Sie im Menü „Edit“ den Eintrag „Find/Replace...“ oder wählen Sie die Tastenkombination „Ctrl+F“. Weitere Informationen erhalten Sie in der Eclipse Hilfe.
- **Syntax Highlighting** - Darunter versteht man das Hervorheben von Java-Schlüsselwörtern. Dies sind reservierte Worte, die einen bestimmten Zweck im Java-Code haben. Sie werden im Hamster-Editor hervorgehoben.
- **Code Assistenz** - Der Hamster-Editor kann Sie beim Schreiben von Hamster-Programmen durch seinen Code-Assistenten unterstützen. Dabei ist dieser abhängig von der Curserpoition und dem was Sie bereits geschrieben haben. Im folgenden werden einige Beispiele gebracht um die Funktionsweise zu verdeutlichen:

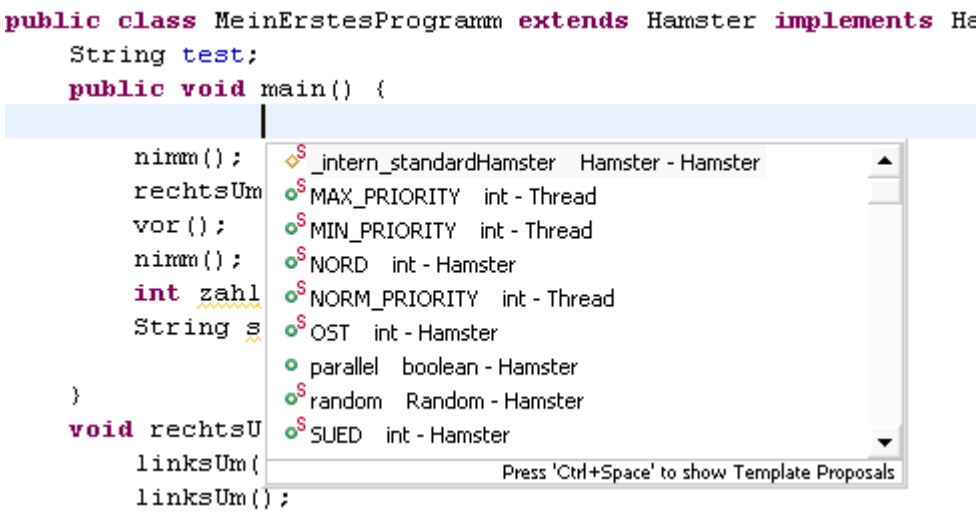


Abbildung 4.18: Code Assistent

1. Gehen Sie in die Methode `void main()`, gehen Sie in eine freie Zeile und rufen Sie den Code-Assistenten durch die Tastenkombination „Ctrl+Space“ auf (sh. Abbildung 4.18). Es wird ihnen eine Liste angezeigt, wobei als erstes die lokalen Variablen, dann die globalen Variablen und schließlich die Methoden aufgeführt werden. Diese sind alle alphabetisch sortiert. Hinter den lokalen Variablen stehen die jeweiligen Datentypen und bei den globalen Variablen stehen zusätzlich die Klassen. Bei Methoden steht zunächst der Methodenname, der Rückgabewert und zu welcher Klasse die Methode gehört. Für Sie sind in dieser Liste nur die globalen Variablen von ihrer eigenen Klasse (Hamster-Programm indem Sie sich befinden) und die Methoden der Klasse `Hamster`, der Klasse `Territorium` und ihrer eigenen Klasse interessant. Wählen Sie die Variable oder Methode aus, die Sie verwenden wollen und drücken Sie die Taste „Enter“ um ihre Auswahl zu übernehmen. Alternativ geht auch Doppelklick mit der linken Maustaste auf das zu wählende Element in der Liste.
2. Gehen Sie wieder in der Methode `void main()` in eine leere Zeile und rufen Sie den Code-Assistenten auf. Durch Schreiben der von ihnen gewünschten Variable/Methode schränkt sich die Liste auf die bereits geschriebenen Buchtstaben weiter ein.
3. Sie können auch die Anfangsbuchstaben der Variable oder Methode, die Sie schreiben wollen, schreiben und dann den Code-Assistenten aufrufen. Ist die Methode/Variable bereits eindeutig, so wird diese vervollständigt. Ist sie nicht eindeutig so wird die bereits eingeschränkte Liste angezeigt.
4. Wenn Sie bereits objektorientiert programmieren, so wird nach dem Schreiben eines Objektbezeichners eines bestimmten komplexen Datentyps Plus eines Punkts der Code-Assistent geöffnet, welcher nun eine Liste mit den Variablen und Methoden dieses Objekts zeigt. Beispiel: es gibt ein Objekt vom Typ `Hamster` mit dem Objektbezeichner `paul`. Wenn Sie nun `paul.` schreiben, so wird Ihnen die Liste der möglichen Variablen und Methoden angezeigt.
5. Der Code-Assist kann auch dazu verwendet werden, Templates zu verwenden. Da-

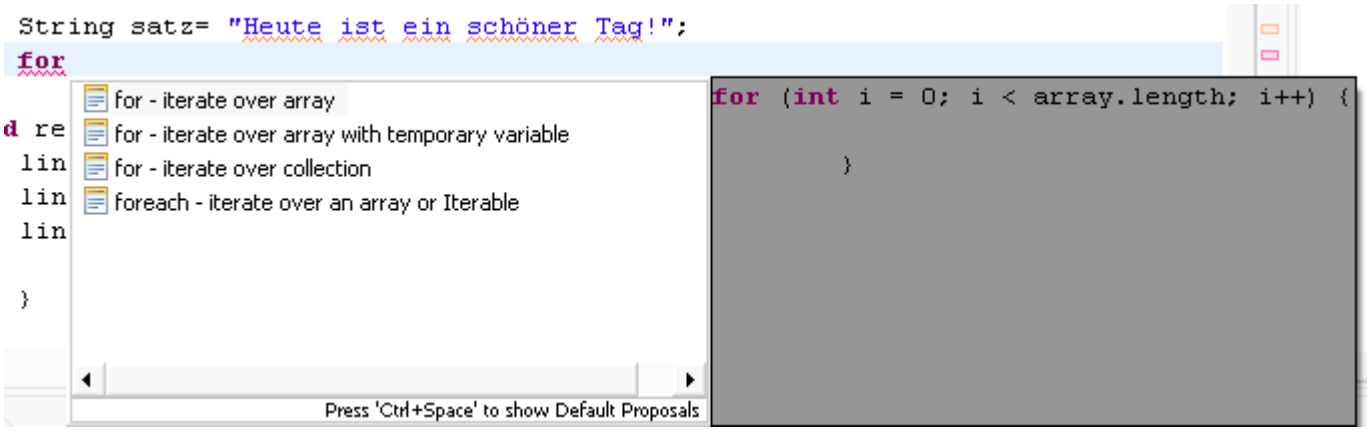
String satz= "Heute ist ein schöner Tag!";
for
for - iterate over array
for - iterate over array with temporary variable
for - iterate over collection
foreach - iterate over an array or Iterable
for (int i = 0; i < array.length; i++) {
}
Press 'Ctrl+Space' to show Default Proposals

Abbildung 4.19: Code Assistent: Templates

bei handelt es sich um häufig verwendete Programmierkonstrukte, wie Schleifen oder bedingte Verzweigungen. Dazu müssen Sie ein Teil des Konstrukts schreiben und dann zweimal hintereinander die Tastenkombination „Ctrl+Space“ wählen. Sie bekommen eine Liste der passenden Templates angezeigt. (sh. Abbildung 4.19) Bsp: Schreiben Sie: `if` und wenden Sie zweimal die Tastenkombination „Ctrl+Space“ an. wählen Sie aus der Liste den Eintrag „if“. Es wird nun der Rumpf einer „if“-Verzweigung generiert. Sie müssen nur noch die Bedingung für „Condition“ einsetzen.

Es gibt unter anderem für die folgenden Schlüsselwörter Templates: if, for, do, while, switch, try

6. Für objektorientierte-Hamster-Programmier: Bisher wurde der Code-Assistent in Methoden ausgeführt. Führen Sie ihn direkt im Klassenrumpf aus, so ändert sich die Liste, denn Sie können nun zum einen Getter- und Setter-Methoden für globale Variablen generieren lassen und Sie können aus der Liste alle Methoden auswählen die geerbt wurden und sich den Code zum überschreiben dieser Methoden generieren lassen.

   Eine andere Möglichkeit sich Getter- und Setter-Methoden zu gloablen Variablen zu erzeugen und geerbte Methoden zu überschreiben besteht darin, dass man in dem Kontextmenü des Hamster-Editors den Eintrag „Source“ (Alt+Shift+S) wählt und dort die entsprechenden Einträge wählt.

- **Hervorheben von gleichen Bezeichnern** - Um gleiche Bezeichner im Code zu identifizieren reicht es den Bezeichner mit der linken Maustaste zu selektieren. Es werden alle anderen gleichen Bezeichner hervorgehoben (sh. Abb. 4.20).
- **Überprüfung des Codes auf syntaktische Fehler zur Laufzeit** - Der Hamster-Editor ist in der Lage ihren Code während Sie ihn eingeben auf syntaktische Fehler hin zu überprüfen und bei Fehlern Ihnen dies in Form von rot unterstrichelten Codeteilen und Fehlermarkern am linken Rand des Editors mitzuteilen, ohne explizit den Code kompilieren zu müssen (sh. Abbildung 4.21). In den „Problems View“ werden die Fehler jedoch erst übernommen, wenn die Datei explizit gespeichert und somit kompiliert wurde.

```
public void main() {
    nimm();
    rechtsUm();
    linksUm();
    vor();
    nimm();
    int zahl = 356;
    String satz= "Heute ist ein schöner Tag!";
}
void rechtsUm() {
    linksUm();
    linksUm();
    linksUm();
    }
```

Abbildung 4.20: Hervorgehobener Code

```
18          linksUm();
19          vor();
20          nimm();
21          vor
22      }
23
24      void rechtsUm() {
25          linksUm();
```

Abbildung 4.21: Syntaktische Fehlerüberprüfung zur Laufzeit

- Finden von Referenzen und Deklarationen - Der Hamster-Editor ist in der Lage Referenzen eines Methodenaufrufs oder eines Objektes innerhalb eines Projekts oder des Workspaces zu finden. Beispielsweise kann man `linksUm()` im Hamster-Editor selektieren und dann in dessen Kontextmenü den Eintrag "References→Projekt" wählen um alle Hamster-Programme im Hamster-Projekt zu finden, die diesen Hamster-Befehl benutzen.
  Das Finden von Deklarationen funktioniert genauso, nur das die Klasse und die Zeile gesucht wird, in der beispielsweise eine Methode oder ein Objekt definiert wurden. Zuständig dafür ist der Eintrag „Declarations" im Kontextmenü des Hamster-Editors
- **Anzeigen von Javadoc-Kommentaren** - Verfügt eine Methode über einen Javadoc-Kommentar in der Deklaration der Methode, so wird dieser als Tooltip angezeigt, wenn man mit der Maus über einem Methodenaufruf dieser Methode verweilt.
- **Zur Deklaration springen** - Um zur Deklaration eines Objektes oder einer Methode zu springen kann man die „Ctrl"-Taste gedrückt halten und mit der linken Maustaste auf das gewünschte Objekt oder Methodenaufruf klicken.
- **Weitere Funktionen** - Der Hamster-Editor verfügt noch über eine Reihe weiterer Funktionen, die beim Programmieren hilfreich sein können. Diese werden in der Eclipse Hilfe im Rahmen des Java Editors näher erläutert.

## 4.4 Verwalten und Editieren von Hamster-Territorien

Um die Grundlagen von Hamster-Territorien und des Territorium-Editors zu verstehen, ist es hilfreich, dass Sie den Abschnitt 4.1.4 durchlesen. Dieser erläutert unter anderem, dass nur ein Territorium gleichzeitig geöffnet sein kann und was es mit dem Standard-Territorium auf sich hat.

### 4.4.1 Verändern der Größe des Hamster-Territoriums

Abbildung 4.22: Hamster Terrain Wizard

Will man die Größe des Hamster-Territoriums ändern, so wird man um das erstellen eines neuen Territorium nicht herumkommen, denn es gibt im Hamster-Simulator-Plugin nicht die Möglichkeit die Größe von bereits bestehenden Hamster-Territorien zu ändern. Um ein neues Territorium einer bestimmten Größe zu erstellen muss man den Hamster-Territorium-Wizard öffnen. Dies erfolgt analog zum öffnen des „Hamster Projekt Wizards“ und des „Hamster File Wizards“ und kann im Abschnitt 4.2.1 nachgelesen werden.

In der Abbildung 4.22 ist der „Hamster Terrain Wizard“ dargestellt, welcher für die Erzeugung neuer Territorien verantwortlich ist. In diesem müssen Sie als Benutzer das Hamster-Projekt und gegebenenfalls den Unterordner angeben, in dem das Hamster-Territorium erstellt werden soll. Dazu können Sie den „Browse“-Button wählen, welcher den „Folder-Selection“-Dialog öffnet, in welchem Sie das gewünschte Hamster-Projekt und Verzeichnis wählen können. Als nächstes müssen Sie im „Hamster Terrain Wizard“ den Namen des Territoriums angeben, wobei das angegebene Territorium im Hamster-Projekt bzw. in dem jeweiligen Unterverzeichnis

noch nicht existieren darf. Als letztes müssen Sie die Größe des neuen Territoriums angeben und den Wizard mit drücken des „Finish“-Buttons beenden. Es wird nun automatisch ein neues Territorium im gewählten Hamster-Projekt bzw. dessen Unterverzeichnis erzeugt und im Territorium-Editor geöffnet. Sie können dieses nun nach ihren Wünschen bearbeiten.

### 4.4.2 Platzieren des Standard-Hamsters im Hamster-Territorium

Abbildung 4.23: Territorium Actionset

Um den Standard-Hamster auf einer beliebigen Kachel im Territorium zu platzieren, wählen Sie bitte im Actionset des Territorium-Editors, in der globalen Toolbar, den „Hamster“-Button (fünfte von links in Abb. 4.23). Sie haben nun das Hamster-Werkzeug ausgewählt und können nun bestimmen, auf welcher Kachel der Hamster platziert werden soll, indem Sie mit der linken Maustaste auf die Kachel klicken. Dabei kann der Standard-Hamster nicht auf Kacheln platziert werden, die bereits durch Mauern versperrt sind. Das Hamster-Werkzeug bleibt solange ausgewählt, bis Sie nochmals den Hamster-Button drücken, um es zu deaktivieren oder bis ein anderes Werkzeug ausgewählt wird. Mit dem Hamster-Werkzeug kann nur der Standard-Hamster auf Kacheln platziert werden. Alle anderen Hamster werden explizit im Hamster-Programm erzeugt.

### 4.4.3 Setzen der Blickrichtung des Standard-Hamsters

Die Blickrichtung des Standard-Hamsters können Sie mit dem „90° nach links drehen“-Button (sechster Button von rechts in Hamster Actionset - Abb. 4.23) verändern. Wenn Sie diesen Button drücken wird der Hamster um 90° nach links gedreht und seine Blickrichtung ändert sich beispielsweise von „Nord“ nach „West“. Bei diesem Button handelt es sich nicht um ein Werkzeug, somit bleibt das vorher aktive Werkzeug weiterhin aktiv!

### 4.4.4 Abfragen und Festlegen der Körneranzahl im Maul des Standard-Hamsters

Abbildung 4.24: Körner im Maul des Standard-Hamsters setzen

Die Körneranzahl die der Standard-Hamster im Maul hat können Sie abfragen und ändern, indem Sie den fünften Button von rechts (Hamster mit Körnern) in Abbildung 4.23 betätigen. Es öffnet sich der in Abbildung 4.24 gezeigt Dialog, der zum einen die Körneranzahl zeigt, die sich aktuell im Maul des Standard-Hamsters befindet und in dem Sie zum anderen eine neue Anzahl an Körnern setzen können.

### 4.4.5 Platzieren von Körnern auf Kacheln des Hamster-Territorium

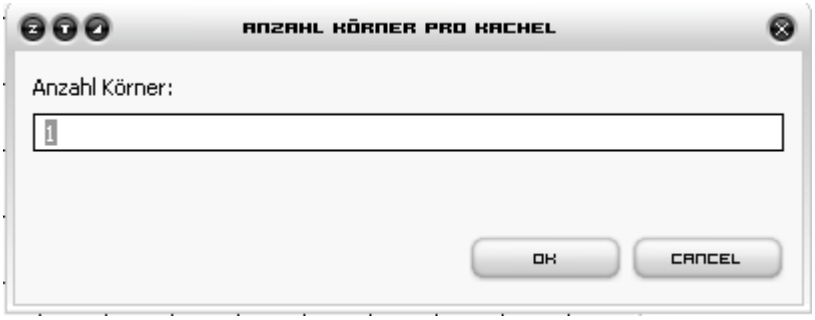


Abbildung 4.25: Anzahl der Körner pro Kachel setzen

Wenn Sie Körner auf Kacheln verteilen möchten, müssen Sie das Korn-Werkzeug aktivieren, indem Sie im Territorium-Actionset (sh. Abb 4.23) den dritten Button von links wählen. Zum deaktivieren des Korn-Werkzeugs müssen Sie den Button erneut wählen oder ein anderes Werkzeug wählen. Haben Sie das Korn-Werkzeug aktiviert, so können Sie durch klicken der Maustaste, auf der entsprechenden Kachel, ein oder mehrere Körner setzen. Dazu erscheint nachdem Sie die Maustaste losgelassen haben der in Abbildung 4.25 gezeigte Dialog, welcher die Eingabe einer Anzahl verlangt. Diese muss aus positiven Zahlen bestehen, wobei maximal 12 Körner pro Kachel angezeigt werden. Es können aber durchaus mehr Körner auf einer Kachel platziert werden.

Wenn Sie auf mehreren Kacheln gleichzeitig dieselbe Anzahl an Körnern platzieren wollen, so müssen Sie das Korn-Werkzeug selektieren und auf die Start-Kachel mit der Maustaste klicken, die Maustaste gedrückt halten und zu der Ziel-Kachel mit der Maus gehen. Wenn Sie dort die Maustaste loslassen erscheint der oben besprochene Dialog, welcher die Anzahl der Körner pro Kachel erfragt. Diese Anzahl wird auf allen überdeckten Kacheln gesetzt und im Territorium-Editor entsprechend angezeigt.

Sie können Körner nicht auf Kacheln platzieren, die durch Mauern versperrt sind.

### 4.4.6 Platzieren von Mauern auf Kacheln des Hamster-Territorium

Um Mauern zu Platzieren gehen Sie analog zum Setzen von Körnern vor. Aktivieren Sie das Mauer-Werkzeug in dem Territorium Actionset (sh. sechster Button von links in Abb. 4.23). Klicken Sie mit der Maustaste auf die Kachel, auf der Sie eine Mauer setzen wollen oder halten Sie die Maustaste gedrückt und gehen Sie von der Start- zur Ziel-Kachel, falls Sie mehrere Mauern gleichzeitig setzen wollen, wobei auf allen überdeckten Felder eine Mauer

gesetzt wird. Mauern können auf allen Kacheln gesetzt werden, die nicht durch den Standard-Hamster belegt sind.

### 4.4.7 Löschen von Kacheln des Hamster-Territorium

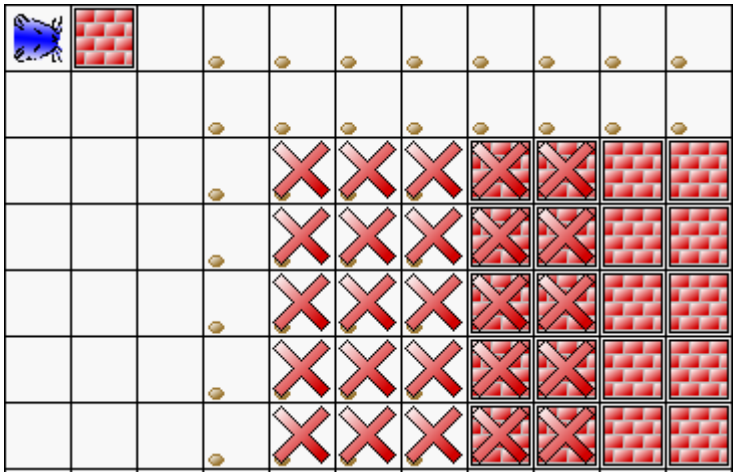

Abbildung 4.26: gleichzeitiges Löschen des Inhalts von mehreren Kacheln

Zum Löschen des Inhalts einer Kachel aktivieren Sie das Lösch-Werkzeug (vierter Button von links im Territorium Actionset (sh. Abb. 4.23)) und klicken Sie auf die Kachel dessen Inhalt Sie löschen wollen. Sie können auch die Maustaste gedrückt halten um den Inhalt mehrerer Kacheln gleichzeitig zu löschen (sh. Abb. 4.26).

### 4.4.8 Abspeichern eines Hamster-Territoriums

Um das Hamster-Territorium zu Speichern müssen Sie den Territorium-Editor selektieren und dann im Menü „File“ den Eintrag „Save As“ wählen. Es erscheint der in Abbildung 4.27 gezeigte „Save As“-Dialog, in welchem Sie zunächst das Hamster-Projekt oder einen Unterornder in einem Hamster-Projekt wählen müssen. Als zweites müssen Sie einen Namen für ihr Territorium angeben unter dem Sie es speichern wollen. Dazu ersetzen Sie den Standard-Namen „defaultTerritorium.ter“ durch einen von ihn gewählten Namen, wobei es wichtig ist, dass Sie die Dateiendung „*.ter“ mit hin schreiben.

Hamster-Territorien können nur explizit auf die oben beschriebenen Weise gespeichert werden. Wird ein Territorium geöffnet oder neu erzeugt so wird das bisher im Territorium-Editor bearbeitete Territorium ohne Nachfrage ersetzt. Ebenso wird das neue Territorium auch sofort zum Standard-Territorium, da es das letzte Territorium ist, das bearbeitet wurde.

Abgespeichert werden die Größe des Territoriums, Standort und Blickrichtung des Standard-Hamsters, sowie Informationen über die Einzelnen Kacheln, d.h. ob Sie mit Mauern oder Körner belegt sind und wenn ja mit wie vielen.

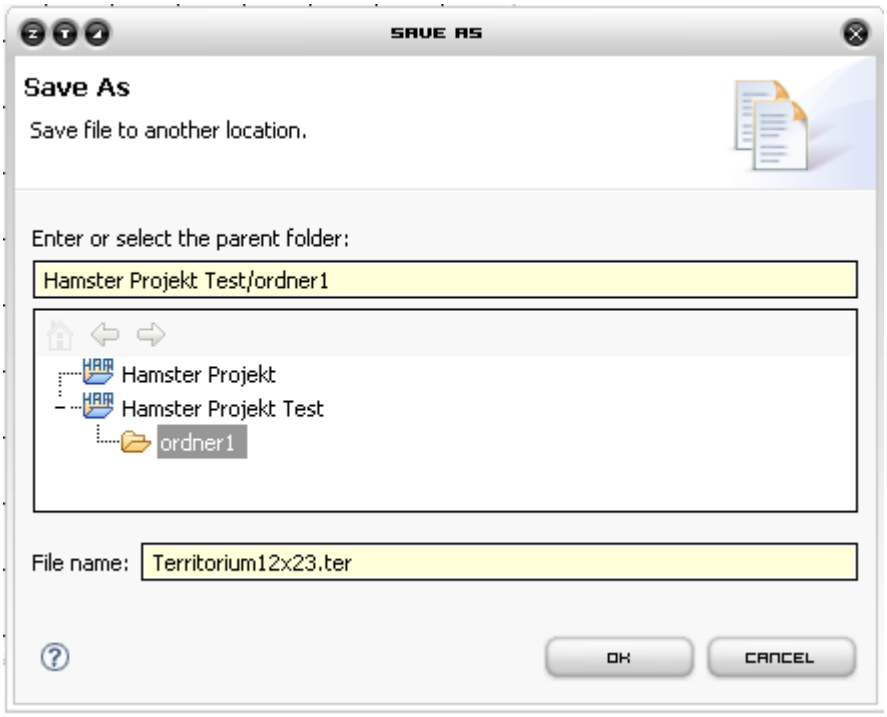


Abbildung 4.27: Speichern eines Hamster Territoriums

### 4.4.9 Wiederherstellen eines abgespeicherten Hamster-Territoriums

Abgespeicherte Hamster-Territorien werden wie Hamster-Programme und Hamster-Klassen auch im „Navigator View“ im jeweiligen Hamster-Projekt dargestellt und können durch Doppelklick mit der linken Maustaste oder durch den Eintrag „open“ im Kontextmenü der Territoriumdatei geöffnet werden. Es ist auch möglich die Territoriumdatei mit einem anderen Editor als dem Territorium-Editor zu öffnen, da es sich dabei um eine reine Text-Datei handelt. Allerdings wird hier davon abgeraten, da zum Bearbeiten dieser Datei genaue Kenntnisse über dessen Struktur vorhanden seien müssen. Diese werden hier jedoch nicht näher erläutert. Ändert man die Territoriumdateien trotzdem mit einem anderen Editor so kann es bei fehlerhafter Bearbeitung vorkommen, dass der Territorium-Editor die Datei nicht mehr öffnen kann.

### 4.4.10 Umbenennen eines abgespeicherten Hamster-Territoriums

Um ein Hamster-Territorium umzubenennen, kann man im Kontextmenü der Territoriumdatei im „Navigator View“ den Eintrag „Rename“ wählen. Alternativ ist auch das selektieren der Territoriumdatei möglich und dann das wählen des Eintrags „Rename“ im Menü „File“ oder das wählen der Taste „F2“. Nun geben Sie den neuen Namen mit Dateiendung ein und bestätigen dies mit der „Enter“-Taste.

### 4.4.11 Löschen und Verschieben und Kopieren von Hamster-Territorien

Das Löschen, Verschieben und Kopieren von Hamster-Territorien bzw. Territoriendateien funktioniert analog zu dem Löschen, Verschieben und Kopieren von Hamster-Programmen und Hamster-Klassen bzw. deren Hamsterdateien. Das Löschen ist in Abschnitt 4.3.4 beschrieben, das Verschieben in Abschnitt 4.3.6 und das Kopieren in Abschnitt 4.3.7.

### 4.4.12 Verändern der Größendarstellung des Hamster-Territoriums

Die Größendarstellung des Hamster-Territoriums können Sie mit den „Zoom-hinein“- und „Zoom-hinaus“-Button (erster und zweiter Button im der Territorium Actionset (sh. Abb. 4.23)) verändern. Dabei wird die Darstellung des Territoriums durch betätigen des „Zoom-hinein“-Button vergrößert und durch Betätigen des „Zoom-hinaus“-Buttons verkleinert.

## 4.5 Kompilieren von Hamster-Programmen

Hamster-Programme und auch Hamster-Klassen müssen in der Regel nicht explizit kompiliert werden. Die Kompilierung einer Hamsterdatei erfolgt bei der Speicherung dieser Dateien durch den Hamster-Editor und es werden die für die Simulation notwendigen ausführbaren Class-Dateien angelegt. Sollte es syntaktische Fehler im Code geben so werden diese nach dem Speichern der entsprechenden Datei im „Problems View“ aufgeführt und die zugehörigen Fehler-Marker am Rand des Hamster-Editors angezeigt.

Ist diese implizite Kompilierugsmethode im „Eclipse SDK“ ausgeschaltet (kein Häkchen beim Eintrag „Build Automatically“ im Menü „Project“), so können Sie das ganze Hamster-Projekt kompilieren, indem Sie den Hamster-Editor oder das Hamster-Projekt selektieren und im Menü „Project“ den Eintrag „Build Projekt“ wählen. Eine zweite Möglichkeit gezielt die Hamsterdatei und nicht das ganze Projekt zu kompilieren besteht darin, im Kontextmenü der Hamsterdatei den Eintrag „Hamsterdatei→ Kompilieren“ zu wählen (sh. auch Abschnitt 4.8.4), wobei darauf zu achten ist, dass die Hamsterdatei gespeichert ist!

## 4.6 Ausführen von Hamster-Programmen

Im Hamster-Simulator-Plugin sind die imperativen und die objektorientierten Hamster-Programme ausführbar. Anders als in früheren Versionen des „Hamster Simulators“, muss zu jedem Hamster-Programm das gestartet werden soll eine so genannte „Hamster-Launchconfiguration“-Datei angelegt werden, welche unter anderem den Namen der Hamsterdatei und das zugehörige Hamster-Projekt speichert. Diese Dateien enden mit der Dateiendung „*.launch“ und werden in der Regel in einem für Sie unzugänglichen Teil des Workspaces aufbewahrt. Allerdings können Sie dies auch beeinflussen, was im folgenden noch erklärt wird. Um eine solche „Hamster-Launchconfiguration“-Datei zu erzeugen müssen Sie das Fenster „Create, manage, and run configurations“ öffnen. Dazu haben Sie folgende Möglichkeiten:

- Sie wählen im Menü „Run“ den Eintrag „Runöder „Debug“‘.

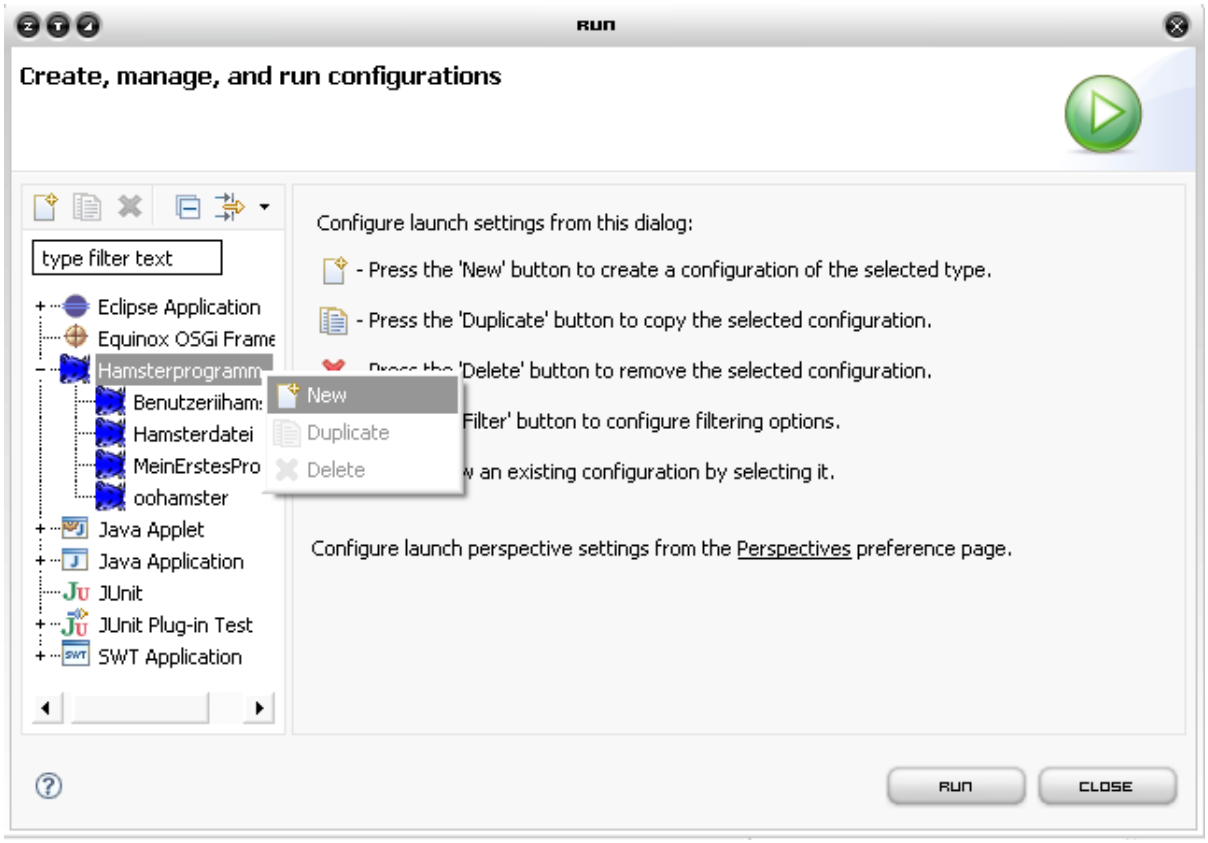


Abbildung 4.28: „Create, manage, and run configurations“-Fenster

- Sie wählen im Drop-Down Menü des Start- oder Debug-Buttons (Launch-Actionset vgl. Abschnitt 4.1.1) den Eintrag „Run“ bzw. „Debug“
- Sie wählen im Kontextmenü eines Hamster-Programms den Eintrag „Run As→ Run“ bzw. „Debug As→ Debug“
- Sie wählen im Kontextmenü des Hamster-Editors den Eintrag „Run As→ Run“ bzw. „Debug As→ Debug“

Es erscheint das in Abbildung 4.28 dargestellte „Create, manage, and run configurations“-Fenster, welches alle Lauchconfigurations verwaltet und diese startet. Auf der linken Seite sehen Sie die verfügbaren Start-Typen die ihr „Eclipse SDK“ unterstützt, wobei „Hamsterprogramme“ einer dieser ist. Wenn Sie mit der linken Maustaste „Hamsterprogramme“ selektieren, so bekommen Sie auf der rechten Seite eine Übersicht der verfügbaren Optionen (ist unterschiedlich in den Eclipse-Versionen; hier für 3.2). Beispielsweise können Sie dort beeinflussen, in welcher Perspektive die Hamster-Programme gestartet werden sollen.

Um eine neue Hamster-Launchconfiguration anzulegen gehen Sie ins Kontextmenü des Eintrags „Hamsterprogramme“ und wählen Sie den Eintrag „new“ aus. Die rechte Seite verändert sich nun und zeigt den „Hamster Main Tab“ (sh. Abb. 4.29), in welchem Sie den Namen der Hamster-Launchconfiguration angeben müssen das Hamster-Projekt und den Namen des Hamster-Programms. Haben Sie eine Hamster-Programm oder einen Hamster-Editor selektiert bevor Sie das „Create, manage, and run configurations“-Fenster geöffnet haben, so werden

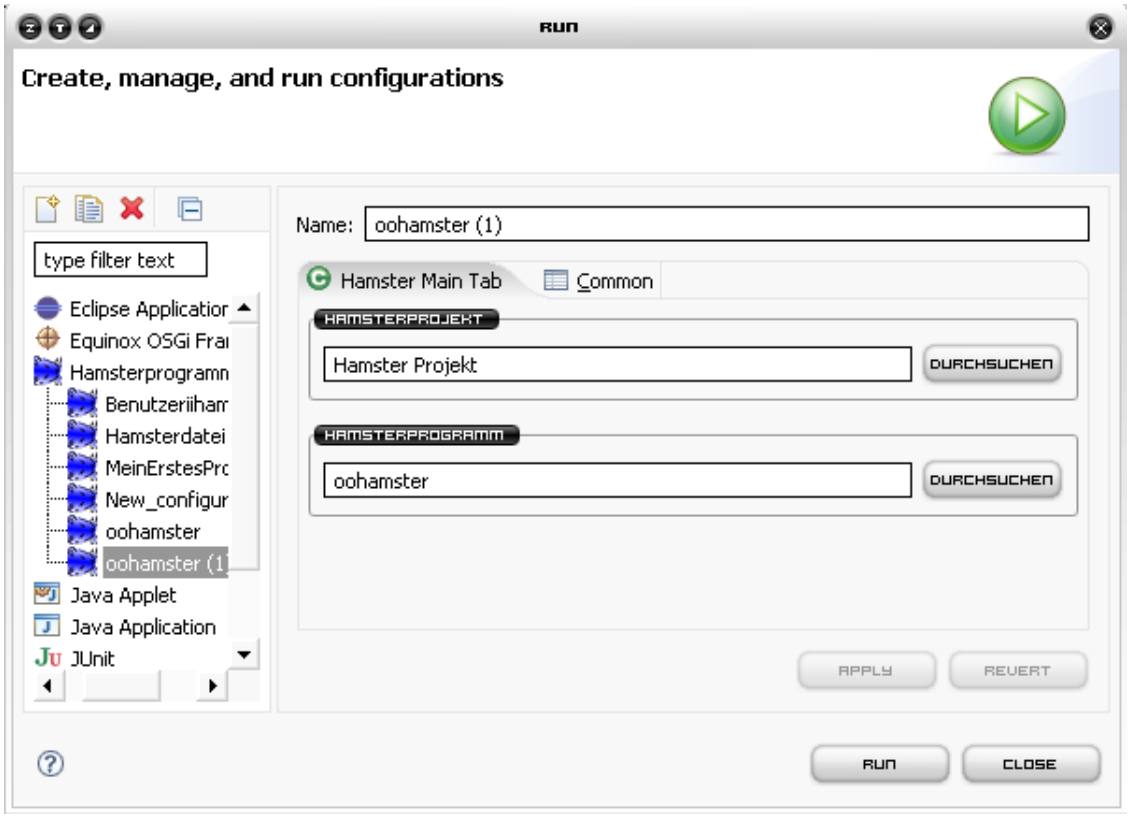


Abbildung 4.29: „Create, manage, and run configurations“-Fenster

automatisch die Daten dieses Hamster-Programms übernommen. Müssen Sie das Hamster-Projekt oder das Hamster-Programm explizit angeben, so können Sie das über die Dialoge machen die nach betätigen der „Browse“-Buttons erscheinen oder manuell hinschreiben. Dabei ist darauf zu achten, wenn Hamster-Programme in Unterordnern liegen auch diese im Feld „Hamsterprogramm“ in der Form „unterordner.Hamsterprogramm“angegeben werden müssen. Anschließend können Sie das „Create, manage, and run configurations“-Fenster Verlassen, indem Sie auf den „Close“-Button klicken oder auf den „Run“- bzw. „Debug-“Button um das Fenster zu schließen und Hamster-Programm im jeweiligen Modus zu starten.

Vorhandene Hamster-Launchconfigurations können im „Create, manage, and run configurations“-Fenster durch selektieren geöffnet und verändert werden, wobei mit dem „Apply“-Button die Änderungen übernommen werden können.

Im „Common Tab“, der neben dem „Hamster Main Tab“ verfügbar ist können Sie zusätzliche Einstellungen für die Launchconfiguration vorgenommen werden. Dazu gehört zum einen die Angabe in welchem Projekt die *.launch Datei gespeichert werden soll, wenn diese verfügbar gemacht werden soll. Zum anderen, ob die Hamster-Launchconfiguration im Favoriten Menü aufgenommen werden soll und des weiteren, ob der Start im Hintergrund ablaufen soll. Daneben können Sie auch noch weitere Einstellungen vornehmen, die jedoch von geringem Interesse für Sie sein dürften.

Das Hamster-Simulator-Plugin bietet Ihnen aber auch einen einfacheren Weg ihr Hamster-Programm zu starten. Dies geschieht über so genannte Hamster-Launchconfiguration-Shortcuts,

welche automatisch bereits bestehende Hamster-Launchconfigurations aufrufen oder sofern es zu ihrem Hamster-Programm noch keine Hamster-Launchconfiguration gibt automatisch neue erzeugen und diese für Sie starten. Auch hier gibt es mehrere Möglichkeiten ein Hamster-Programm zu starten:

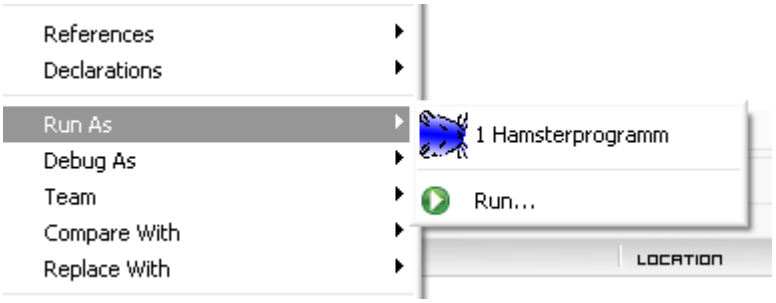


Abbildung 4.30: Run As - Menü

- Sie können den Hamster-Editor, in dem Sie ihr Hamster-Programm bearbeiten, selektieren und in dessen Kontextmenü den Eintrag „Run As→ Hamsterprogramm“ bzw. „Debug As→ Hamsterprogramm“ (sh. Abb. 4.30) wählen.
- Sie können im „Navigator View“ das die Hamsterdatei mit dem Hamster-Programm selektieren und in dessen Kontextmenü den Eintrag „Run As→ Hamsterprogramm“ bzw. „Debug As→ Hamsterprogramm“ wählen. Dabei muss das Hamster-Programm nicht im Hamster-Editor geöffnet sein.
- Sie können aber auch die Hamsterdatei oder den Hamster-Editor selektieren und im Menü „Run“ den Eintrag „Run As→ Hamsterprogramm“ bzw. „Debug As→ Hamsterprogramm“ wählen.
- Haben Sie ein Hamster-Programm kürzlich gestartet, so existiert eine Hamster-Launchconfiguration, welche in der „Run History“ bzw. „Debug History“ aufgeführt wird. Diese erreichen Sie Über das Menü „Run“ und den Eintrag „Run History“ bzw. „Debug History“ oder über die „Drop Down“-Menüs der Start- und Stopp-Buttons im Launch-Actionset.

Programme die im „run“-Modus gestartet werden können über den „Stopp“-Button angehalten werden. Das Pausieren und Fortsetzen der Ausführung von Hamster-Programmen ist nur im „debug“-Modus möglich. Während der Ausführung sind alle Buttons des Territorium-Actionsets nutzbar!

Aus programmiertechnischen Gründen ist es im Hamster-Simulator-Plugin notwendig die Hamster-Dateien für die Zeit der Ausführung zu erweitern. Dazu wird bei imperativen Hamster-Programmen ein so genannter Konstruktor sowie eine zusätzliche statische `main`-Methode mit dem Parameter `String[] args` hinzugefügt. Bei objektorientierten Hamster-Programmen wird nur die die statische `main`-Methode hinzugefügt. Diese Erweiterungen verschwinden nach der Ausführung wieder und dürfen bei der Ausführung besonders im „debug“-Modus nicht verändert werden.

## 4.7 Debuggen von Hamster-Programmen

Wie ein Hamster-Programm im „debug“-Modus ausgeführt wird, wurde im vorherigen Abschnitt besprochen. Die Ausführung eines Hamster-Programms im „debug“-Modus dient prinzipiell dem aufspüren von syntaktischen Fehlern im Programm-Code. Daher ist es auch möglich die Ausführung an einer beliebigen Stelle anzuhalten , fortzusetzen und gegebenenfalls den Programmcode Zeile für Zeile auszuführen und bei Bedarf abzuändern ohne das Programm neu starten zu müssen. Dabei bedient sich das Hamster-Simulator-Plugin der bereits bestehenden Konzepte des „Eclipse SDK“ so das beispielsweise die „Hot Swap“-Funktionalität oder das Konzept der „Breakpoints“ verwendet werden kann. Ebenso wird zum debuggen von Hamster-Programmen der „Debug-View“ (vgl. Abschnitt 4.1.8), der „Variables View“, der „Breakpoints View“ und der „Expressions View“ verwendet.

Grundsätzlich funktioniert der Start eines Hamster-Programms so, dass eine neue „Java Virtual Machine“ (Client-VM) gestartet wird, in das Hamster-Programm abläuft und welche mit dem Hamster-Simulator-Plugin, das in der „Java Virtual Machine“ (Server-VM) des „Eclipse SDK“ läuft, über die Standardeingabe und Standardausgabe kommuniziert. Dabei wird in der Client-VM im „debug“-Modus ein so genanntes „Debug Target“ erzeugt, welches wiederum mehrere „Threads“ beinhaltet, die für die Umsetzung des Hamster-Programms sorgen. Dabei führt der „main-Thread“ das eigentliche Hamster-Programm aus.

Abbildung 4.31: Fortsetzen-, Pause- und Stopp-Button

Ist die Ausführung des Hamster-Programms im „debug“-Modus gestartet, so werden die in Abbildung 4.31 dargestellten Pause und Fortsetzen Buttons zusätzlich aktiviert, um das Hamster-Programm zu Pausieren und Fortzusetzen. Diese Buttons werden sowohl in der globalen Toolbar als auch in der lokalen Toolbar des „Debug Views“ dargestellt und sind abhängig von der jeweiligen Selektion im „Debug View“. Hat man im „Debug View“ nichts selektiert und betätigt während der Programm-Ausführung den Pause-Button der globalen Toolbar, so ist das „Debug Target“ implizit selektiert und wird veranlasst zu Pausieren, was ebenfalls dazu führt das die im „Debug Target“ enthaltenden Thread angehalten werden und somit auch der „main-Thread“. Durch betätigen des Fortsetzen-Buttons wird das „Debug Target“ und dessen „Threads“ fortgesetzt. Es ist auch möglich einzelne „Threads“ zu Pausieren, was für parallele Hamster-Programme wichtig sein dürfte (dritter Band). Dazu muss man im „Debug View“ den „Thread“ selektieren, welchen man pausieren will und einen der Pause-Buttons betätigen.

Um imperative oder objektorientiert Hamster-Programme zu debuggen müssen Sie das „Debug Target“ oder den „main-Thread“ pausieren. Selektieren Sie nun, wie in Abbildung 4.32 zu sehen, die letzte Methode ihres Hamster-Programms (in der Regel unterhalb von Methoden der Klasse „Hamster“ oder „Territorium“) im „Callstack“ (vgl. Abschnitt 4.1.8) und es erscheint der Hamster-Editor mit der selektierten Methode. Sie haben nun die Möglichkeit die Programmausführung schrittweise fortzuführen, indem Sie die Pfeiltasten „Schritt hinein“ (Step Into), „Schritt hinüber“ (Step Over) und „Schritt hinaus“ (Step Return) wählen. Dabei können Sie mit „Schritt hinein“ in den einen Methodenaufruf hineinspringen, mit „Schritt hinüber“ bleiben Sie in der aktuellen Methode und führen den Methodenaufruf aus, ohne selbst in die Methode hinein zu springen und mit „Schritt hinaus“ verlassen Sie die aktuelle Methode

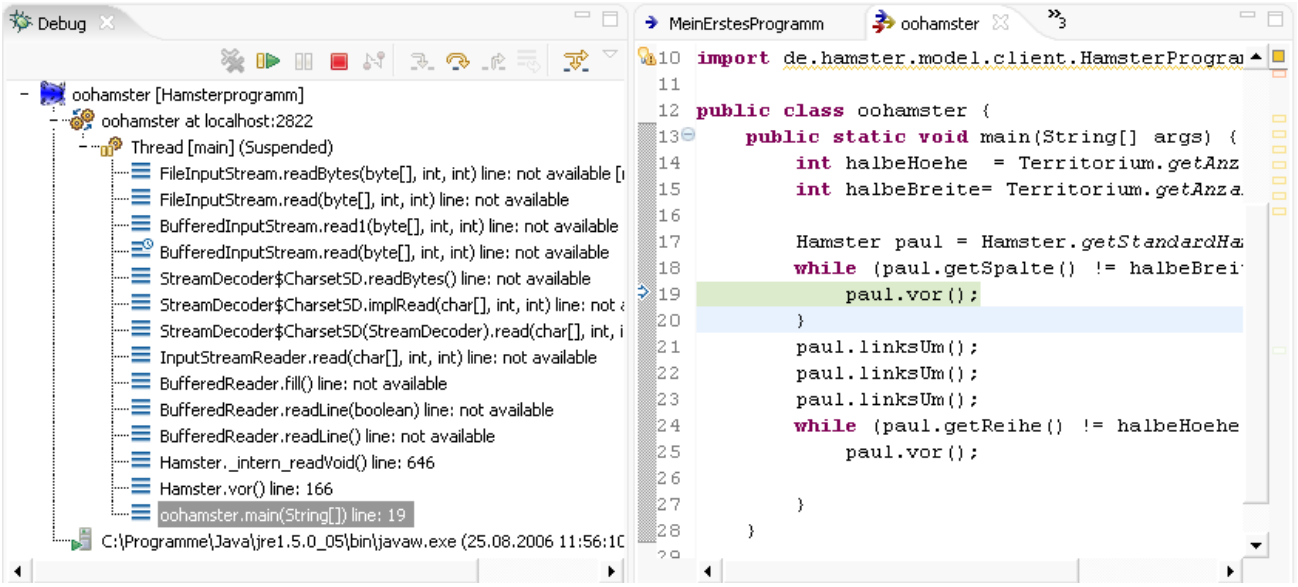


Abbildung 4.32: Debug View und Hamster-Editor

und kehren zu der Methode zurück, welche die so eben verlassene Methode aufgerufen hat. Das Debuggen wirkt sich auch auf die Simulation im Territorium-Editor aus denn diese ist ebenso pausiert bzw. wird schrittweise ausgeführt.

Der Variables View (vgl. 4.1.9) zeigt ihnen während des Debuggens, je nach selektierter Methode, die lokalen und globalen Variablen der Klasse zu der die Methode gehört. Ein weiteres nützliches Werkzeug beim Debuggen stellt der „Expressions View“ dar. Er kann über das Menü „Window“ und den Eintrag “Show View → Other → Debug → Expressions“ geöffnet werden. In dessen Kontextmenü kann man mit dem Eintrag „Add Watch Expression“ einen neuen Ausdruck hinzufügen, der ausgewertet wird und ein Ergebnis liefert, ohne das eigentliche Hamster-Programm zu beeinflussen. Zum Beispiel kann man als Ausdruck eine Variable festlegen und es wird einem, wie im „Variables View“ deren Belegung angezeigt oder man kann auch einen Ausdruck der Form `variable==variable2 && variable2!=variable3`, welcher dann zu `true` oder `false` ausgewertet wird. Dabei muss die Programmausführung hinter der Deklaration der Variablen pausiert worden sein. Ein Beispiel des „Expression Views“ ist in der Abbildung 4.33 dargestellt. Es zeigt als erstes den Rückgabewert des Aufrufs einer statischen Methode der Klasse `Territorium`, danach den Wert einer lokalen Variablen und danach den Rückgabewert der Methode `vornFrei()`, wobei „paul“ ein objektorientierter Hamster ist. Schließlich wird der Rückgabewert der statischen Methode mit der lokalen Variablen verglichen und das Ergebnis ist `false`. Weitere Informationen zum Expressions-View werden in der Eclipse Hilfe gegeben.

Das Hamster-Simulator-Plugin unterstützt das Konzept der „Breakpoints“. Dabei handelt es sich um Marker die am linken Rand des Hamster-Editors gesetzt werden können und die Ausführung des Hamster-Programms im „debug“-Modus an genau der markierten Stelle pausieren können. Um einen Breakpoint im Hamster-Programm zu setzen, müssen Sie, wie in Abbildung 4.34 gezeigt, das Kontextmenü der Markerleiste aufrufen, wobei sie dies genau in der Zeile tun müssen, in der Sie den Breakpoint setzen wollen. Wählen Sie den Eintrag

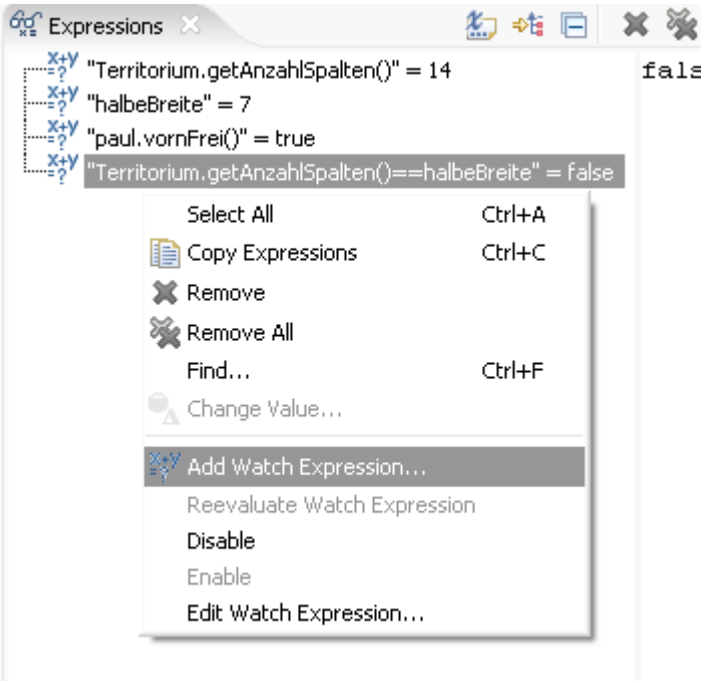


Abbildung 4.33: Expressions View während der Ausführung eines Hamster-Programms

„Toggle Breakpoint“ um einen Breakpoint in der Zeile zu setzen. Es erscheint das Symbol eines blauen Kreises links von der Zeilennummer, welches angibt, dass der Breakpoint aktiv ist. Um einen Breakpoint zu löschen, öffnen Sie nochmals das Kontextmenü und wählen den Eintrag „Toggle Breakpoint“. Mit dem Eintrag „Disable Breakpoint“ können Sie den Breakpoint deaktivieren und es erscheint dann das Symbol eines weißen Kreises. Wenn Sie nun das Hamster-Programm im „debug“- Modus ausführen werden Sie feststellen, dass die Ausführung in der Zeile des Breakpoints pausiert wird und Sie nun die Ausführung zeilenweise oder komplett fortsetzen können. In der Hamster-Debug-Perspektive wird der „Breakpoints View“ (oben rechts) angezeigt, welcher Aufschluss über alle gesetzten, aktivierten und deaktivierten Breakpoints gibt. Diese können hier gelöscht, deaktiviert und aktiviert werden und Sie können zu dem Breakpoint springen. Weitere Informationen finden Sie in der Eclipse Hilfe

Des Weiteren wird das Konzept der Code Änderung während des Ausführens im „debug“-Modus („Hot Swap“) vom Hamster-Simulator-Plugin unterstützt. Dazu können Sie im pausierten Hamster-Programm den Code ändern und müssen anschließend das Programm speichern (kompilieren). Wird für die Veränderung „Hot Swap“ unterstützt, so beginnt die Methodenausführung der Methode, in der Sie sich befinden, von vorne. Wird für die Veränderung kein „Hot Swap“ unterstützt, so fragt Sie der in Abbildung 4.35 gezeigte Dialog, ob Sie die Ausführung des Hamster-Programms fortsetzen wollen, abbrechen wollen oder neu starten wollen. Es wird empfohlen das Programm neu zu starten.

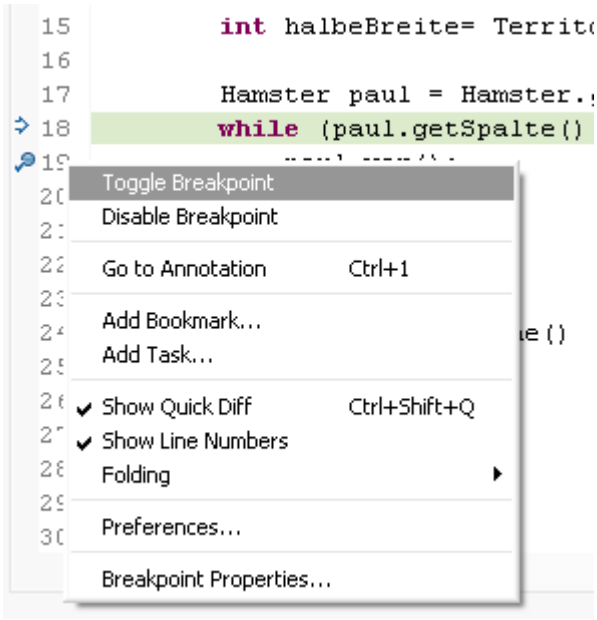


Abbildung 4.34: Deaktivieren eines Breakpoints

## 4.8 sonstige Funktionen des Hamster-Simulator-Pugins

### 4.8.1 Umwandeln von Hamsterdateien ins „*.ham“-Format

Da das Hamster-Simulator-Plugin andere Arten von Hamsterdateien als der „Hamster Simulator“ in der Version 2.2 verwendet und trotzdem die Abwärtskompatibilität der Hamster-Programme gewährleistet werden soll, gibt es die Möglichkeit die Hamsterdateien des Hamster-Simulator-Plugins in das „*.ham“-Dateiformat zu überführen. Dazu müssen Sie im „Navigator View“ die Hamsterdatei selektieren und in dessen Kontextmenü den Eintrag „Hamsterdatei→ *.iham,*.oham,*.cham –> *.ham“. Die „*.ham“-Datei erscheint im „Navigator View“ im gleichen Verzeichnis wie die Ausgangsdatei und kann von dort kopiert oder verschoben werden.

### 4.8.2 Umwandeln von „*.ham“-Dateien in Hamsterdateien

Das Hamster-Simulator-Plugin ist in der Lage, die Hamsterdateien des „Hamster Simulators“ in der Version 2.2 zu verarbeiten. Diese können direkt mit dem Hamster-Editor geöffnet werden oder explizit in Hamsterdateien des Hamster-Simulator-Plugins umgewandelt werden. Dazu müssen Sie die „*.ham“-Datei selektieren und in deren Kontextmenü den Eintrag „Hamsterdatei→ *.ham –> *.iham,*.oham,*.cham“ wählen. Die Hamsterdatei wird erzeugt und die „*.ham“-Datei wird gelöscht. Die neue Hamsterdatei wird jedoch nicht im Hamster-Editor geöffnet.

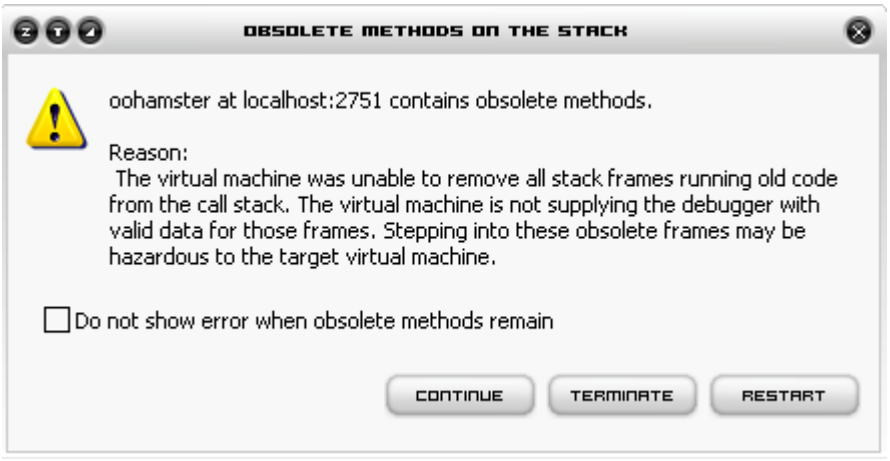
OBSOLETE METHODS ON THE STACK
oohamster at localhost:2751 contains obsolete methods.
Reason:
The virtual machine was unable to remove all stack frames running old code from the call stack. The virtual machine is not supplying the debugger with valid data for those frames. Stepping into these obsolete frames may be hazardous to the target virtual machine.
Do not show error when obsolete methods remain
CONTINUE
TERMINATE
RESTART

Abbildung 4.35: Hamster-Programm neu starten wenn kein Hot Swap möglich

### 4.8.3 Umwandeln von „*.cham“-Dateien in „*.oham“-Dateien

Manchmal ist es notwendig, dass man Hamster-Klassen in Hamster-Programme umwandeln möchte. Dazu können Sie den Eintrag „Hamsterdatei→ *.cham –> *.oham“ wählen, was unter anderem bewirkt, dass die Hamsterdatei ein „main“-Methode erhält und die Dateiendung in *.oham umbenannt wird.

### 4.8.4 Explizites Kompilieren von Hamsterdateien

Sie Haben die Möglichkeit einzelne Hamsterdateien explizit zu kompilieren. Dies ist sinnvoll, wenn das implizite kompilieren von Hamster-Dateien mit dem Speichen im Hamster-Editor ausgeschaltet ist oder die Hamsterdatei mit einem anderen Editor als dem Hamster-Editor bearbeitet wurde. Es werden die zugehörigen Java- und Class-Dateien erzeugt und mögliche Fehler im „Problems View“ angezeigt. Um die Hamsterdatei explizit zu Kompilieren wählen Sie in deren Kontextmenü im „Navigator View“ den Eintrag „Hamsterdatei→ Kompilieren“.

## 4.9 Sonstige Probleme

### 4.9.1 Out of sync Fehler

Dieser Fehler tritt immer dann auf, wenn eine bestimmte Ressource im Workspace nicht synchron mit dem Dateisystem des Betriebssystems ist. In diesem Fall muss die Ressource extern aktualisiert werden, indem Sie die Ressource selektieren und die Taste „F5“ betätigen.

# 5 Noch Fragen?

Wenn Sie noch Fragen zum Hamster-Simulator-Plugin haben sollten, die in diesem Dokument nicht beantwortet wurden, wenden Sie sich bitte an:

- **Matthias Mertens** - Email: matthias.mertens@informatik.uni-oldenburg.de
- **Dr. Dietrich Boles** - Email: boles@informatik.uni-oldenburg.de

# A Anhang

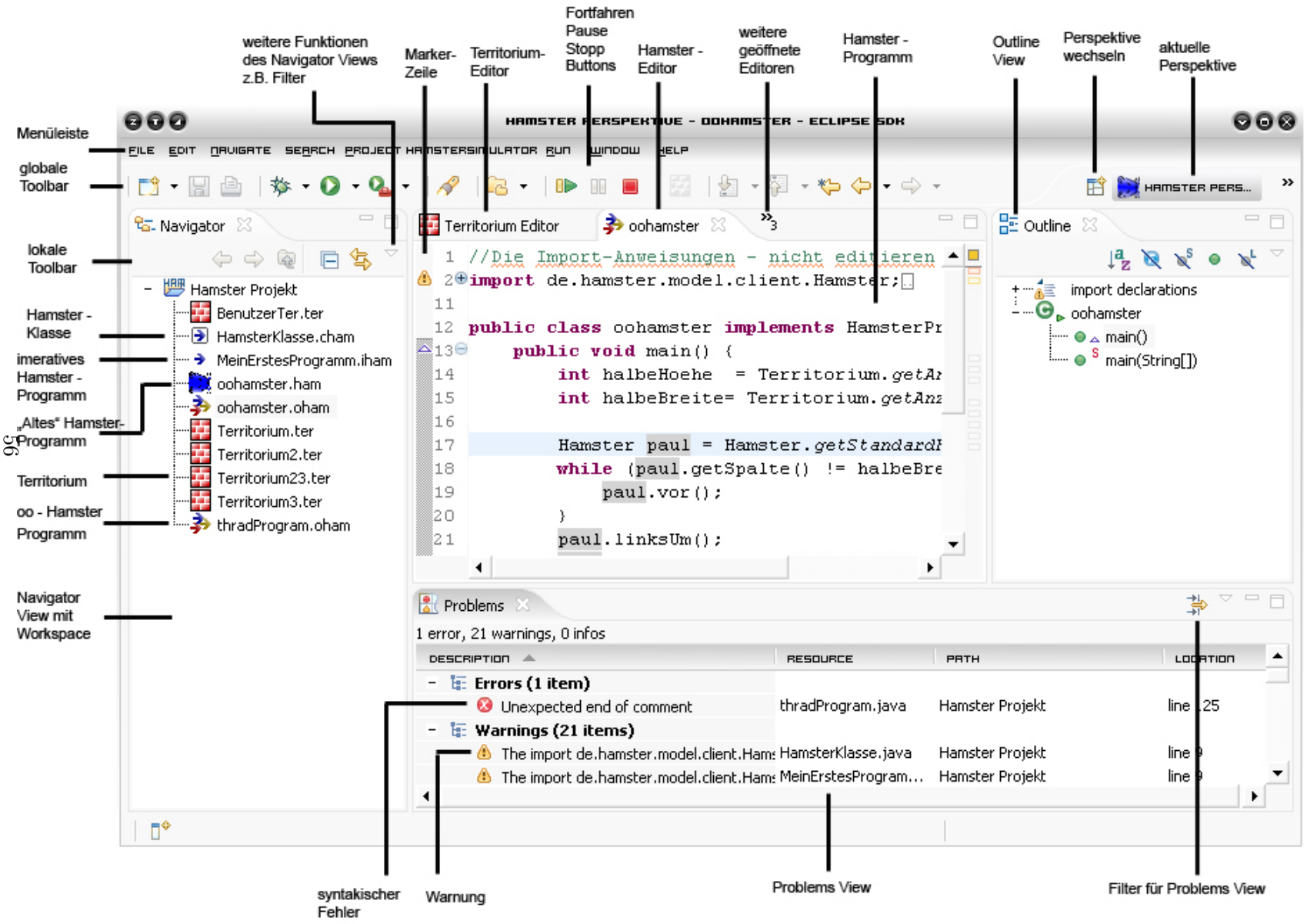


Abbildung A.1: Hamster Perspektive

# Abbildungsverzeichnis


3.1 Hamster-Territorium . . . 12
3.2 Debug View: Callstack . . . 15

4.1 Globale Toolbar . . . 18
4.2 Navigator View . . . 20
4.3 Territorium-Editor . . . 21
4.4 Hamster-Editor . . . 22
4.5 Outline View . . . 23
4.6 Problems View . . . 24
4.7 Debug View . . . 25
4.8 Variables View . . . 26
4.9 Hamster-Projekt-Wizard . . . 27
4.10 Geschlossenes Projekt im „Navigator View“ . . . 28
4.11 Projekt Eigenschaften . . . 29
4.12 „Hamster File Wizard“ . . . 30
4.13 Hamsterdatei im anderen Editor öffnen . . . 31
4.14 „Save As“-Dialog . . . 32
4.15 Schließen eines Editors . . . 35
4.16 Markierter Code . . . 35
4.17 Teile des Menü „Edit“ . . . 36
4.18 Code Assistent . . . 37
4.19 Code Assistent: Templates . . . 38
4.20 Hervorgehobener Code . . . 39
4.21 Syntaktische Fehlerüberprüfung zur Laufzeit . . . 39
4.22 Hamster Terrain Wizard . . . 40
4.23 Territorium Actionset . . . 41
4.24 Körner im Maul des Standard-Hamsters setzen . . . 41
4.25 Anzahl der Körner pro Kachel setzen . . . 42
4.26 gleichzeitiges Löschen des Inhalts von mehreren Kacheln . . . 43
4.27 Speichern eines Hamster Territoriums . . . 44
4.28 „Create, manage, and run configurations“-Fenster . . . 46
4.29 „Create, manage, and run configurations“-Fenster . . . 47
4.30 Run As - Menü . . . 48
4.31 Fortsetzen-, Pause- und Stopp-Button . . . 49
4.32 Debug View und Hamster-Editor . . . 50
4.33 Expressions View während der Ausführung eines Hamster-Programms . . . 51
4.34 Deaktivieren eines Breakpoints . . . 52
4.35 Hamster-Programm neu starten wenn kein Hot Swap möglich . . . 53

*Abbildungsverzeichnis*

A.1 Hamster Perspektive . . . . . 56